Buku Materi Ajar Kuliah
Pemrograman Komputer
Modul 1 – 9

Bertho Tantular, M.Si.

PEMROGRAMAN KOMPUTER

FAKULTAS MATEMATIKA DAN IPA
UNIVERSITAS PADJADJARAN
TAHUN AKADEMIK 2013/2014

KATA PENGANTAR

Bismillahirrahmanirrahiim.

Segala puji dan syukur penulis panjatkan kepada Allah SWT atas segala nikmat dan kuasa-Nya akhirnya modul Pemrograman Komputer dapat diselesaikan. Modul ini merupakan salah satu acuan bagi mahasiswa Jurusan Statistika yang sedang mengikuti praktikum mata kuliah pemrograman komputer. Modul ini merupakan penjelasan singkat penggunaan software Lazarus untuk statistika dengan disertai contoh-contoh. Modul ini terdiri atas sembilan bagian yaitu Bahasa Pemrograman, Objek Pascal dalam Lazarus, Struktur Kondisi, Pengulangan, Variabel Berindeks, Prosedur dan Fungsi, Membuat Grafik, Object Oriented Programming.

Modul ini penulis harapkan dapat membantu mahasiswa dalam melaksanakan Praktikum mata kuliah pemrograma komputer. Penulis menyadari masih banyak kekurangan pada modul ini. Untuk itu penulis mengharapkan adanya kritik dan saran guna menyempurnakan modul ini agar nantinya dapat menjadi manfaat bagi pembacanya,

Akhir kata penulis ucapkan terima kasih banyak kepada semua pihak yang telah membantu penulisan modul praktikum ini.

Penulis,

Bertho Tantular, M.Si.

DAFTAR ISI

DESKRIPSI MATA KULIAH

Mata kuliah Pemrograman komputer adalah mata kuliah yang memberikan penjelasan singkat mengenai bahasa pemrograman Pascal dan penggunaan software Lazarus untuk statistika dengan disertai contoh-contoh. Tujuan mata kuliah ini agar mahasiswa memahami algoritma-algoritma dalam Bahasa Pemrograman Pascal, kondisi, pengulangan, array, prosedur dan fungsi, dan Object Oriented Programming sehingga mampu membuat suatu software statistika menggunakan bahasa pemrograman visual Lazarus.

MODUL 1

BAHASA PEMROGRAMAN

PENDAHULUAN

Dalam modul ini akan dipelajari mengenai pengertian bahasa pemrograman, jenis-jenis bahasa pemrograman, sejarah bahasa Pascal dan perkembangannya hingga ke bahasa pemrograman visual.

Modul ini terdiri atas dua bagian yaitu mengenai bahasa pemrograman ograman Pascal dan Pengenalan Bahasa Pemrograman Lazarus. Dalam bagian pertama akan dijelaskan mengenai pengertian bahasa pemrograman, jenis-jenis bahasa pemrograman, sejarah bahasa pemrograman Pascal, fitur-fitur objek Pascal serta algoirtma dan diagram alir (Flow Chart).

Pada bagian kedua akan dijelaskan mengenai pengenalan software Lazarus, installasi software Lazarus, pengenalan lingkungan kerja Lazarus, membuat dan mengatur area kerja (Form), mengatur properties objek dalam form, pengenalan istilah-istilah dalam software Lazarus seperti *event, method, properties*.

Bagian 1

BAHASA PEMROGRAMAN PASCAL

A. Pengertian Bahasa Pemrograman

Program komputer adalah suatu himpunan dari perintah yang memberitahukan kepada komputer apa yang harus dilakukan. Agar perintah tersebut dapat dimengerti oleh komputer, perintah tersebut perlu ditulis dalam bahasa yang dimengerti komputer yang disebut Bahasa Pemrograman.

Fungsi utama dari bahasa pemrograman adalah mengubah perintah-perintah dalam bahasa manusia menjadi bahasa yang dimengerti oleh mesin, atau lebih singkatnya disebut bahasa mesin sehingga perintah tersebut dapat dilaksanakan oleh komputer. Secara umum bahasa pemrograman dibagi menjadi tiga bagian yaitu:

1. Bahasa Tingkat Rendah

   yaitu bahasa pemrograman yang paling dekat dengan bahasa mesin, biasanya hanya bermain dengan angka 0 dan 1 saja. Contohnya: *Assembly*, *Turbo Assembler.*

2. Bahasa Tingkat Menengah

   yaitu bahasa pemrograman yang menggunakan perintah-perintah sederhana dengan satu kata saja. Keunggulannya adalah kecepatan proses untuk bahasa ini cukup baik. Contohnya: *C, C++, Visual C++*

3. Bahasa Tingkat Tinggi

   yaitu bahasa pemrograman yang paling dekat dengan bahasa manusia, dengan kata lain menggunakan perintah-perintah dalam bahasa manusia khususnya Bahasa Inggris. Kekurangannya adalah kecepatan prosesnya kurang baik. Contohnya: *Pascal, Basic, COBOL, Fortran, Basic, Delphi*.

Untuk memudahkan dalam pembuatan program telah sejak lama dirancang suatu pemrograman yang terstruktur. Arti kata terstruktur adalah suatu cara pemrograman yang hanya mempunyai satu masukan alur program, dan

menghindari penggunaan lebih dari satu masukan. Salah satu program yang telah lama dirancang untuk pemrograman terstruktur adalah *Pascal.*

B. Bahasa Pascal dan Perkembangannya

Bahasa PASCAL pertama kali dikembangkan pada awal tahun 70-an oleh NICLAUS WIRTH di Technical University, Zurich – Swiss. Nama PASCAL diambil dari nama seorang ahli matematika bangsa Perancis, yaitu BLEISE PASCAL yang telah berjasa menemukan alat hitung mekanis pertama didunia pada abad ke-17.

Pada Awalnya bahasa pemrograman ini diperkenalkan dengan tujuan untuk menjelaskan masalah pemrograman komputer bagi mahasiswa yang belajar pemrograman komputer. Ternyata dalam waktu singkat, nahasa pemrograman ini menjadi salah satu bahasa yang sangat populer dikalangan universitas, sehingga menjadi julukan sebagai bahasa universitas. Mulai dari awal perkembangannya hingga saat ini banyak sekali jenis bahasa pemrograman ini, masing-masing merupakan hasil pengembangannya, antara lain :

UCSD Pascal

Microsoft Pascal

Apple Pascal

Turbo Pascal

Diantara versi-versi yang ada, Turbo Pascal merupakan versi yang sangat populer saat ini. Bahasa pemrograman ini termasuk kategori “High Level Language”. Instruksi-instruksi yang digunakan dalam bahasa pemrograman ini sangat sistematis dan terstruktur.

Bahasa Pascal merupakan bahasa pemrograman pertama yang mendukung object oriented programming (*Object Oriented Programming*) yang pertama. Mulanya Bahasa Pascal dikembangkan oleh ***Apple Computer Company*** pada tahun 1983 tetapi kemudian **Borland** mengembangkannya dengan nama Turbo Pascal. Turbo Pascal merupakan bahasa pemrograman berbasis pada *command line interface*. Turbo Pascal merupakan bahasa yang case insensitive yang berarti penulisan dalam huruf kapital maupun huruf kecil tidak dipermasalahkan.

Objek Pascal secara umum dapat dikatakan sebagai bahasa pemrograman hybrid dalam arti bahasa Pascal mendukung pada Pemrograman berstruktur dan berorientasi objek. (*Structured and Object Oriented Programming*). Bahasa Pascal ini dapat digunakan untuk berbagai aplikasi seperti: aplikasi pengajaran, pengembangan game, aplikasi bisnis, aplikasi internet, aplikasi komunikasi, alat-alat pengembangan dan kernel sistem operasi.

Seiring dengan perkembangan teknologi terkait dengan sistem operasi yang digunakan Borland mengembangkan bahasa pemrograman Pascal agar dapat diadaptasi oleh sistem operasi yang baru. Perkembangan sistem operasi visual seperti Windows, Mac OS dan Linux akhirnya Borland memutuskan untuk membuat pengembangan Bahasa Pascal menjadi Bahasa pemrograman visual dengan nama **Delphi**. Dalam perkembangannnya Delphi menjadi Alat pengembangan aplikasi terbaik (*Rapid Application Development*) saat itu. Versi pertama Delphi di rilis pada tahun 1995 yang kaya akan paket-paket dan sejumlah set komponen yang mendukung pengembangan aplikasi Windows dan database.

Dengan berkembangnaya bahasa pemrograman visual dukungan Borland terhadap bahasa pemrograman berbasis command line interface semakin berkurang. Hal ini dimanfaatkan oleh tim **freepascal** untuk mengembangkan proyek open source untuk membuat aplikasi compiler yang kompatibel dengan Turbo Pascal. Freepascal merupakan bahasa pemrograman yang multiplatform sehingga bisa berjalan untuk berbagai sistem operasi seperti Windows, Linux dan Mac OS. Free Pascal 1.0 pertama kali di rilis pada bulan Juli tahun 2000.

Free Pascal adalah compiler yang kekurangannya adalah tidak mendukung IDE (*Integrated Development Environment*) seperti Delphi. Tim free pascal akhirnya mengembangkan proyek Lazarus yang merupakan IDE dari Free Pascal, yang menampilkan editor kode sumber(source code editor), debugger, berbagai framework dan paket berikut komponen kepustakaan (library) yang mirip dengan Delphi IDE. Lazarus versi 1.0 di rilis pada Agustus tahun 2012, karena sifatnya yang open source banyak sukarelawan yang menuliskan paket dan komponen untuk lazarus dan komunitasnya telah berkembang.

C. Fitur Object Pascal

Object Pascal adalah bahasa pemrograman tingkat tinggi sehingga yang sangat mudah untuk dipelajari oleh siapapun. Compiler bahasa Pascal sangat cepat dan aplikasi-aplikasi yang hasilkan sangat reliabel, cepat dan bisa dibandingkan dengan bahasa C atau C++. Kita dapat menuliskan aplikasi-aplikasi yang besar dengan IDE Delphi tanpa menemui kompleksitas dalam penulisannya.

Masalah yang muncul dalam penggunaan software *Borland Delph*i adalah bahwa untuk dapat menggunakan software tersebut membutuhkan lisensi yang berbayar. Hal ini tentunya menjadi kendala terutama bagi kalangan mahasiswa. Saat ini berbagai upaya telah dilakukan untuk mengatasi kendala tersebut, salah satunya adalah dengan dibuatnya software-software gratis untuk kalangan mahasiswa (*student version*) dan software gratis sama sekali (*freeware*). Software-software seperti itu cukup pesat perkembangannya dalam dekade terakhir karena adanya platform linux yang *open source* dalam arti semua orang tanpa batas dapat mengubah, memodifikasi, mengganti, menggunakan dan mendistribusikannya secara gatis (GNU Public License /GPL). Salah satu software yang mendukung hal tersebut adalah Lazarus.

Lazarus merupakan software berbasis *Pascal* yang dirancang untuk memenuhi berbagai jenis pemrograman grafis. Karenanya Lazarus disebut sebagai "kloning"-nya Delphi tetapi tentu saja dengan lisensi gratis.

D. Algoritma dan Flow Chart

Algoritma merupakan suatu langkah-langkah dalam penyelesaian suatu masalah. Langkah-langkah tersebut dapat ditulis menggunakan notasi apa saja, asalkan mudah dibaca dan dimengerti, karena tidak ada notasi baku dalam penulisan algoritma. Setiap orang dapat membuat aturan penulisan dan notasi algoritmanya sendiri. Algoritma komputasi mestinya dibuat atau disesuaikan dengan bahasa pemragraman yang akan digunakan dalam penyelesaian masalahnya.

Secara umum algoritma terdiri atas tiga bagian yaitu

1. Judul (*header*)

   yaitu bagain teks yang digunakan untuk mendefinisikan nama program

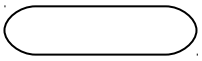
2. Deklarasi (*declaration*)

yaitu bagian yang digunakan untuk mendefinisikan tipe, konstanta, variabel dan fungsi atau prosedur

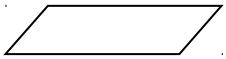
3. Algoritma

yaitu bagian yang berisi perintah-perintah (*statement*) yanng terdiri atas input/output, assignment, sequence (urutan) analisa, kondisi dan pengulangan.

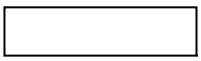
Secara umum algoritma dapat digambarkan dengan menggunakan simbol-simbol yang mudah dibaca dan dimengerti menggunakan diagram allir (FLOW CHART). Suatu program yang cukup kompleks dapat digambarkan menggunakan *Flow Chart* dari program tersebut. Kegunaan dari *Flow Chart* selain untuk memudahkan dalam pembuatan program juga untuk membantu orang lain yang ingin mempelajari program yang kita buat. Beberapa simbol yang digunakan dalam *Flow Chart*



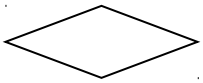
Untuk memulai (*Start*) atau mengakhiri (*End*) program



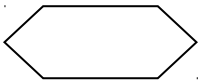
Untuk masukan (*Input*)



Untuk sebuah pernyataan (*Block Statement*)



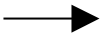
Untuk menyatakan kondisi (percabangan)



Untuk menyatakan deklarasi variabel, konstanta dll.

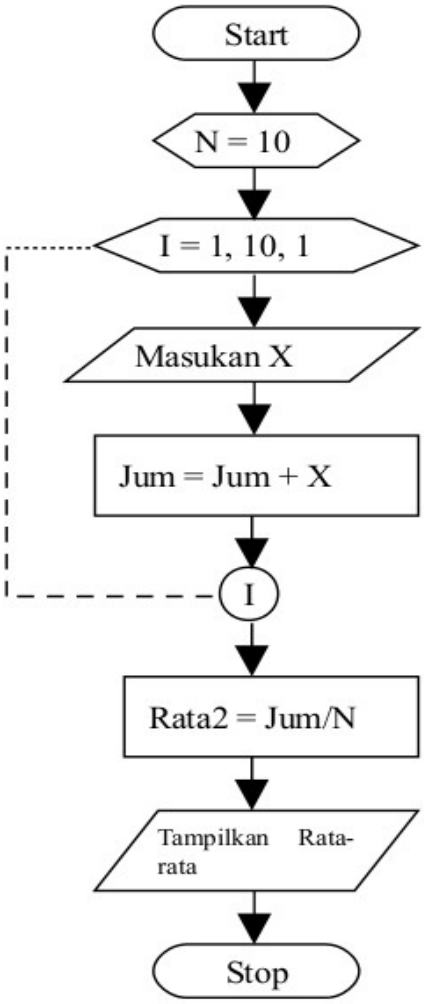
Untuk menyatakan sambungan atau penghubung

Untuk penghubung antar halaman



Untuk penghubung dan arah antar simbol

Selain itu masih banyak simbol-simbol lainnya. Mungkin untuk program sederhana seperti contoh berikut tidaklah penting membuat Flow chart. Tetapi untuk program yang jauh lebih rumit atau kompleks Flow Chart akan sangat berguna. Berikut adalah contoh Flow Chart sederhana untuk menghitung rata-rata



E. Soal-soal

1. Sebutkan bahasa pemograman yang kita ketahui ?
2. Apakah yang dimaksud dengan flow chart ?

3. Mengapa dalam membuat suatu program komputer dianjurkan untuk membuat flow chart terlebih dahulu ?
4. Buatlah flow chart untuk menghitung varians dari populasi dari rumus

Bagian 2

BAHASA PEMROGRAMAN LAZARUS

A. Pengenalan Software Lazarus

Lazarus merupakan perangkat pengembangan aplikasi berbasis pascal dilingkungan Linux. Dalam perkembangannya Lazarus juga mendukung berbagai platform lain seperti Windows, Mac OS dan Solaris.

Dengan menggunakan perangkat lunak ini kita dapat membangun berbagai aplikasi GUI (permainan, multimedia, database dan lain-lain). Dengan pendekatan visual kita dapat menciptakan aplikasi canggih tanpa banyak menuliskan kode.

Banyak bahasa pemprograman yang ada dewasa ini baik yang berbasis text (CLI) maupun grafis (GUI), baik yang berbasis Windows, Linux, Sun Solaris maupun sistem operasi lainnya. Pada bahasan ini yang akan digunakan sebagai kajian yaitu bahasa pemrograman Lazarus. Pertimbangan penggunaan program ini karena Lazarus merupakan pemprograman visual yang sangat hkital dan mudah untuk dipelajari. Lazarus merupakan pengembangan dari Bahasa Pemrograman Pascal yang sangat terkenal pada Pemrograman berbasis DOS.

Lazarus merupakan sebuah software open source dengan free lisensi sehingga kita dapat mendapatkannya secara mengunduhnya gratis **Lazarus IDE** berikut **Free Pascal Compiler**-nya di situs http://lazarus.freepascal.org.

Lazarus tersedia untuk berbagai platform baik Windows, Linux, Sun Solaris maupun Mac OS. Aplikasi yang ditulis dalam Lazarus dapat di re-compiled pada platform lain untuk mendapatkan aplikasi yang dapat dieksekusi pada platform tersebut. Sebagi contoh apabila kita menulis aplikasi menggunakan Lazarus dalam platform Windows, dan kamu menginginkan aplikasi tersebut bisa di eksekusi dalam Linux kita tinggal menyalin (*copy*) kode sumber (*source code*) ke Lazarus dalam Linux kemudian tinggal dicompile.

Lazarus memproduksi aplikasi asli pada setiap sistem operasi, dan tidak membutuhkan library tambahan atau mesin virtual. Oleh sebab itu sangat mudah untuk menyebarkan dan mengekseskusi secara cepat.

B. Instalasi Program Lazarus

Sebelum dapat memulai dan menggunakan Bahasa Pemrograman Lazarus, kita harus mengetahui dulu dukungan software ini dalam platform apa. Untuk platform windows file instalasi Lazarus berekstensi *.exe dengan space sekitar 60MB. Perlu diketahui Bahasa Pemrograman Lazarus menggunakan objek-objek dalam Pascal sehingga kita harus mengetahui perintah-perintah dasar dalam pascal.

Setelah memutuskan penggunaan Versi Lazarus maka langkah selanjutnya adalah menginstalasi Program Lazarus. Proses instalasi program Lazarus cukup mudah seperti pada instalasi program berbasis Windows lainnya, tinggal ikuti petunjuk yang diberikan pada tampilan di layar.

Untuk pengguna Linux kita dapat dengan mudah memperolehnya dalam software repository. Apabila kita menggunakan Ubuntu kita tinggal menuliskan perintah berikut di terminal

```
sudo apt-get install lazarus
```

Apabila kita menggunakan Fedora kita tinggal menuliskan perintah berikut di terminal

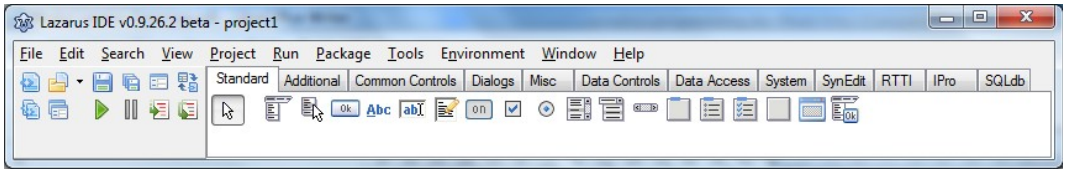
```
yum install lazarus
```

Sesudah Lazarus terinstalasi pada komputer, kita bisa memulai mengenal apa itu Lazarus. Lazarus adalah kompiler/penterjemah bahasa Lazarus (awalnya dari Pascal) yang merupakan bahasa tingkat tinggi sekelas dengan Basic, Fortran dan C. Bahasa Pemrograman Lazarus disebut bahasa prosedural artinya bahasa/sintaknya mengikuti urutan tertentu/prosedur. Ada jenis pemrograman non-prosedural seperti pemrograman untuk kecerdasan buatan seperti bahasa Prolog.

Lazarus termasuk Keluarga Bahasa pemrograman Visual yang berarti perintah-perintah untuk membuat objek dapat dilakukan secara visual. Pemrogram

tinggal memilih objek apa yang ingin dimasukkan kedalam Form, lalu tingkah laku objek tersebut saat menerima event/aksi tinggal dibuat programnya. Lazarus merupakan bahasa pemrogramann berorentasi objek, artinya nama objek, properti dan metode/procedure dikemas menjadi satu kemasan (encapsulate).

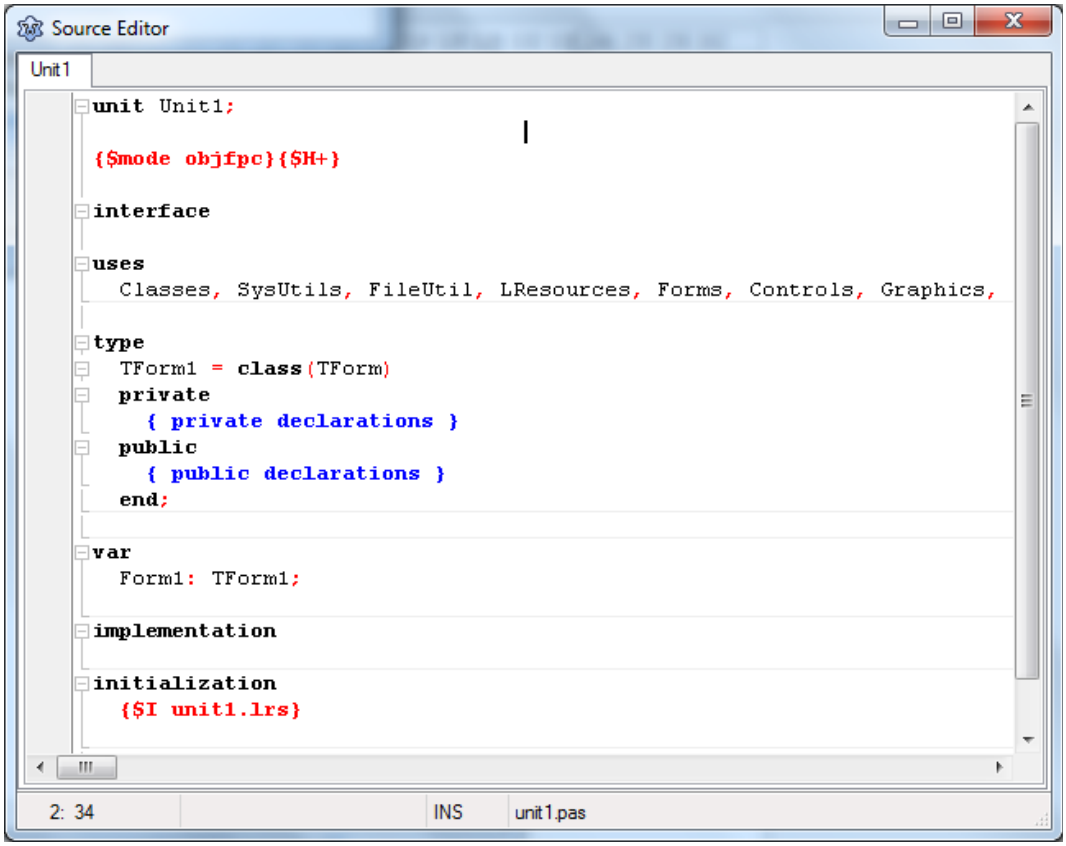
Sebelum mempelajari ketiga struktur pemrograman ada baiknya kenali dahulu tampilan IDE, yang merupakan editor dan tools untuk membuat program Lazarus. Pada IDE akan ditampilkan Form baru yang merupakan aplikasi/program Window yang akan dibuat. Aplikasi/program berbasis windows sering disebut dengan jendela (window). Bagaimana membuat aplikasi berbasis windows (berbasis grafik dan bukan berbasis teks)? Caranya dengan membuat sebuah form. Pada pemrograman berbasis windows, kita akan diperhadapkan pada satu atau beberapa jendela yang nampak dihadapan kita. Jendela ini dalam Lazarus disebut juga dengan form. Pada pemrograman berbasis windows, kita akan diperhadapkan pada satu atau beberapa jendela yang nampak dihadapan kita. Jendela ini dalam Lazarus disebut juga dengan form.



Gambar 1. Jendela Utama Lazarus

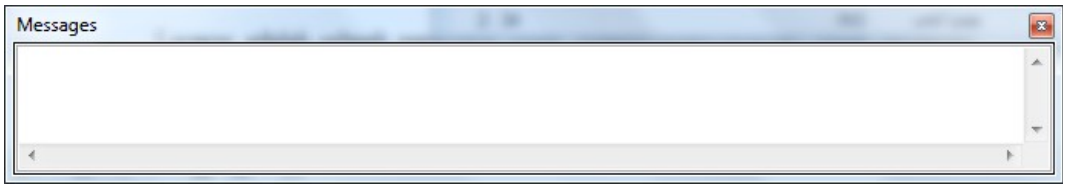
Pada jendela utama (main Windows) ini berisi ikon-ikon untuk mengatur program yang akan dibuat. Selain itu dalam jendela ini juga berisi menu-menu untuk membuka atau menyimpan objek, membuat membuka dan menyimpan projek, menu untuk eksekusi (run), pengaturan penggunaan tolls, menu editing dan menu bantuan (help). Sayangnya Lazarus hanya menyediakan menu bantuan secara online dan tidak menyediakan secara offline.

Pada bagian bawah jendela ini ada tab-tab yang berisi palet-palet yaitu berupa objek-objek yang nantinya akan ditempatkan didalam Form.



Gambar 2. Jendela Editor Sumber (Source Editor)

Pada jendela source editor ini kode-kode program akan dituliskan.



Gambar 3. Jendela Pesan (Messages)

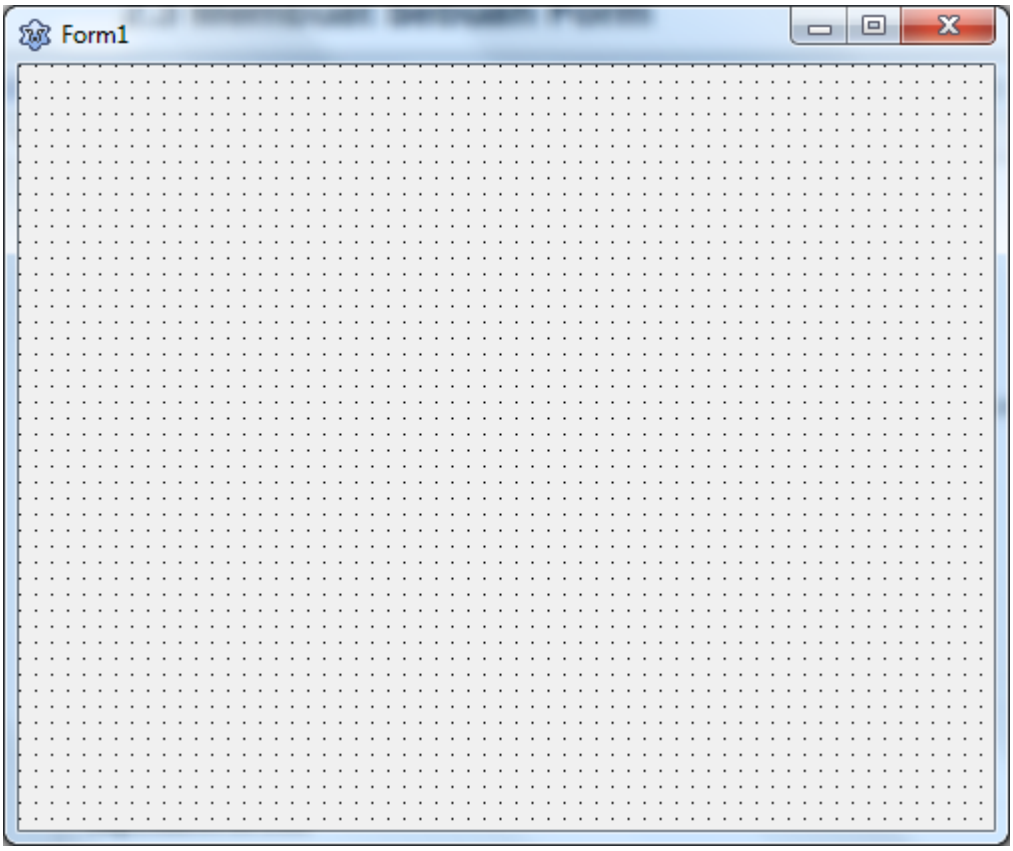
Messages merupakan jendela yang nantinya akan berisi semua pesan kesalahan (error), peringatan (warning) maupun keterangan sukses tidaknya program yang dijalankan.

Jendela lainnya adalah Form yaitu tempat kita membuat program dan Object inspector yaitu jendela untuk mengatur objek yang ditempatkan di dalam Form. Untuk kedua jendela ini akan dijelaskan pada bagian selanjutnya.

Lazarus adalah sebuah perangkat lunak (bahasa pemrograman) untuk membuat program / aplikasi komputer berbasis windows. Lazarus merupakan bahasa pemograman berbasis objek, artinya semua komponen yang ada merupakan objek-objek. Ciri sebuah objek adalah memiliki nama, properti dan method/procedure. Lazarus disebut juga *visual programming* artinya komponen-komponen yang ada tidak hanya berupa teks (yang sebenarnya program kecil) tetapi muncul berupa gambar-gambar.

C. Membuat Sebuah Form

Saat kita pertama kali masuk ke Lazarus, kita akan diperhadapkan ada sebuah form kosong yang akan dibuat secara otomatis. Form tersebut diberi nama **Form1.** Form ini merupakan tempat bekerja untuk membuat antarmuka pengguna.

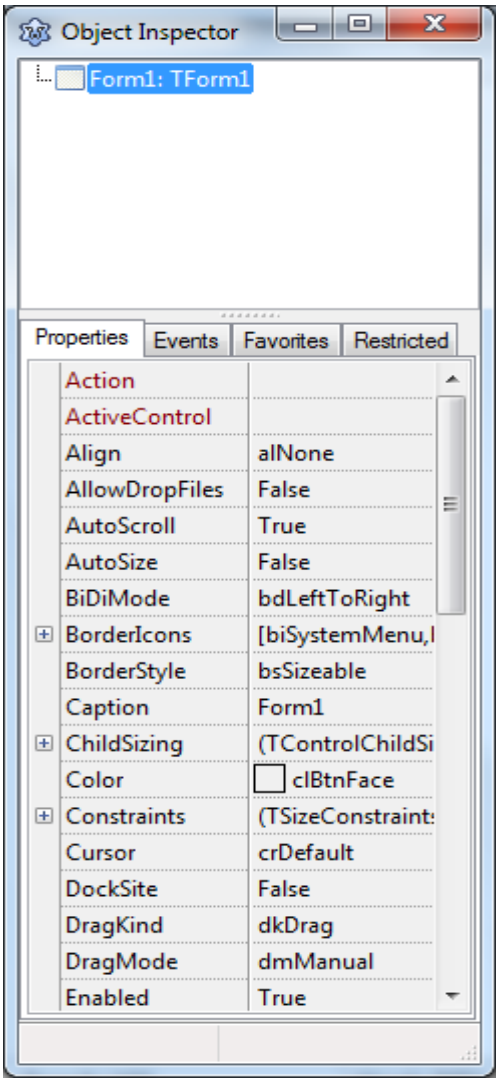


Gambar 4. Form Kosong Pada saat masuk Lazarus

D. Mengganti Nama Form dan Menambahkan Judul

Biasakan sebelum menjalankan program, sebaiknya ganti nama form dan beri judul sesuai program yang kita buat. Lazarus akan secara otomatis memberi nama form1, form2, form3, dst. Nama form tersebut kurang mengandung arti dan akan menyulitkan bila form yang dibuat cukup banyak. Saat membuka Lazarus pertama kali, nampak sebuah jendela **Object Inspector.** Jika tidak muncul pilih menu *View* | *Object Inspector* atau tekan tombol F11.

Pada *Object Inspector* ada dua buah halaman (tab) yaitu Properties dan Events. **Properties** digunakan untuk mengganti properti (kepemilikan) sebuah objek/komponen. Sedangkan **Events** digunakan untuk membuat procedure yang diaktifkan (trigered) lewat sebuah event.



Gambar 5. Tampilan Objeck Inspector Untuk Pengaturan Properti dan even

Semua properti diurutkan berdasarkan alpabetik, dan dapat juga diurutkan berdasarkan kategori. Gantilah judul form dengan Hello melalui properti ***Caption***, sedangkan nama form dengan nama frmHello melalui properti ***Name***. Caption digunakan untuk menyimpan keterangan yang dimunculkan pada form, sedangkan Name digunakan sebagai Nama dari objek tersebut. Isi dari properti Name harus diawali alpabet dan tidak menggunakan spasi atau tanda baca kita sekarang sudah membuat aplikasi form kosong dengan tombol standar window : Minimize, Maximize, dan Close. kita dapat mengubah ukuran form dengan menarik pada bingkai form menggunakan mouse (*drag*=klik tombol kiri mouse, tahan tombol tersebut lalu geser ke kiri/kanan atau atas/bawah). kita dapat memindahkan form dengan meletakkan kursor pada form kemudian menggesernya (drag).

Menyimpan Form

Pada Lazarus ada 3 buah file utama (***.dp**r, ***.pas** dan ***.dfm**).

1) *.**dpr** adalah file proyek yang dibuat berisi program kecil untuk :
   - Mendefinisikan Unit yang ada dalam file proyek
   - Menginisialisasi data
   - Membangun form
   - Menjalankan aplikasi

```
uses
Forms,
Unit1 in 'Unit1.pas' {Form1};
begin
Application.Initialize;
Application.CreateForm(Tform1, Form1);
Application.Run;
end.
```

2) *.**pas** adalah unit-unit (*pascal code file*), yang merupakan kode program yang kita tuliskan untuk mengeksekusi even. Unit kode program pascal ini bisa terdiri satu atau banyak file

3) *.**dfm** adalah file definisi Form (*special pseudo code file*), bisa terdiri satu atau banyak file

```
object Form1: Tform1
Left = 200
Top = 108
Width = 696
Height = 480
Caption = 'Form1'
Font.Charset = DEFAULT_CHARSET
Font.Color = clWindowText
Font.Height = -11
Font.Name = 'MS Sans Serif'
Font.Style = [ ]
PixelsPerInch = 96
TextHeight = 13
object Button1: Tbutton
Left = 176
Top = 116
Width = 75
Height = 25
Caption = 'Button1'
TabOrder = 0
end
end
```

Catatan:

Setiap Form (.dfm) harus memiliki sebuah Unit (.pas), tetapi kita dapat memiliki Unit tanpa sebuah Form (hanya kode saja). Jika ingin melihat kode tersebut kita dapat mengklik kanan mouse, lalu pilih ***VIEW AS TEXT*** *atau tekan tombol* ***Alt-F12.*** *Sebaiknya kita tidak mengubah isi code tersebut, karena akan menyebabkan masalah serius. Tunggu saat kita sudah memahami maksud kode tersebut. Untuk kembali ke bentuk form, pilih* ***VIEW AS FORM*** *atau tekan tombol* ***Alt- F12*** *kembali.*

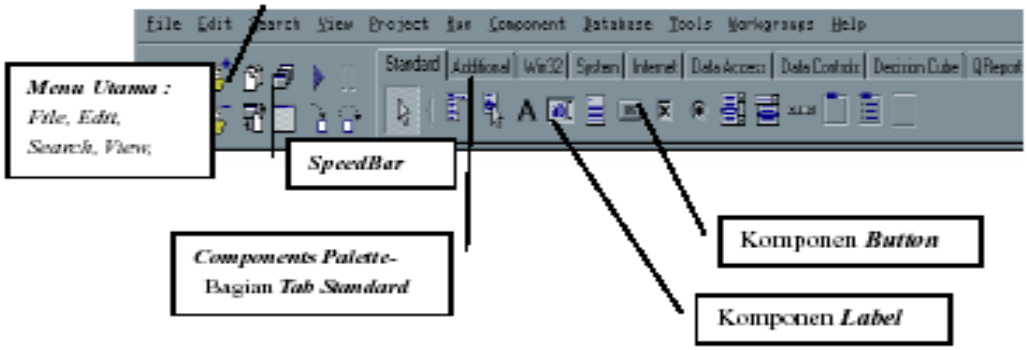
Untuk menyimpan program dalam Lazarus dapat dilakukan dengan memilih submenu **Save Project** atau **Save Project As** pada menu File, dan Lazarus akan menanyakan nama file source code untuk unit (*.pas) dan nama file proyeknya (*.dpr). Beri nama file form dengan **lat1.pas** dan project **latihan1.lpi** Sesudah disimpan, jalankan program dengan menekan tombol F9 atau pilih menu *Run* | *Run*.

E. Menempatkan Komponen pada Form

Karena Lazarus merupakan bahasa pemrograman visual, maka komponen-komponen akan nampak pada layar. kita tinggal menempatkan komponen yang diinginkan pada form. Ada empat cara menempatkan komponen pada form. Misal kita memilih komponen ***Button*** pada ***Components Palette*** bagian ***Standard Page***. kita dapat memilih salah satu langkah berikut:

- Klik pada kompenen tersebut, pindahkan kursor ke form, sambil menekan tombol kiri mouse (drag komponen dan geser pada form) atau
- Pilih komponen (klik komponen yang diinginkan) pada ***Components Palette*** kemudian klik pada form dimana komponen itu akan diletakkan.
- Klik ganda pada komponen yang diinginkan, maka komponen tersebut akan ditambahkan pada form

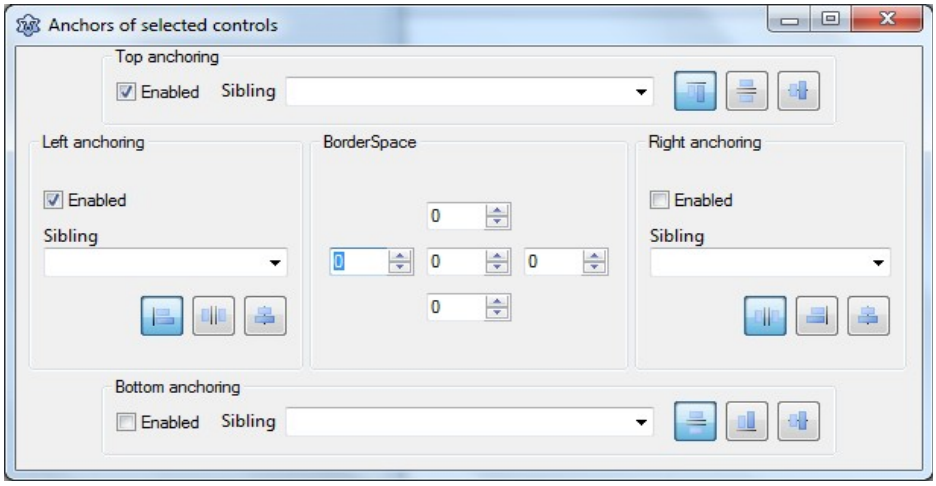
- kita dapat menggunakan *Copy* dan *Paste* bila ingin membuat komponen yang sama yang sudah ada pada form. Caranya Shift-Klik kiri pada komponen yang ada di form, lalu pilih menu *Copy* (Ctrl-C) kemudian pilih menu *Paste* (Ctrl-V).



Gambar 6. Komponen Yang ada di Lazarus

F. Mengatur Tataletak Komponen

Pada form ini hanya ada satu button, mungkin ada di bagian tengah form. kita dapat mengatur letak komponen tersebut dengan menggesernya. Bila kita ingin merapihkan pilih menu **View | Anchor Editor**, maka muncul sebuah Toolbox Anchor dengan ikon perapihan



Gambar 7. Kontrol untuk Objek

Dengan toolbox ini kita dapat merapikan beberapa komponen sekaligus, caranya buat fokus beberapa komponen, lalu klik icon pada toolbox yang diinginkan. Untuk mengetahui arti icon tersebut gerakan mouse pada tombol tersebut, lalu akan muncul penjelasan singkat kegunaan icon tersebut atau lihat Help (tekan F1). kita bisa mempelajarinya sendiri. Cukup mudah!. Langkah yang penting adalah mengubah nama dan keterangan kompenen button tersebut. Ikuti bagian ini

G. Mengubah Nilai Properti

Ubah nilai properti **Caption** menjadi *Katakan Hello* dan nilai properti **Name** menjadi *btnHello*. Langkah ini mirip dengan mengubah nama dan keterangan sebuah form. Setiap komponen sebaiknya diberi nama yang memiliki arti dan diawali oleh jenis komponennya. Misal nama dari form Hello adalah "frmHello" atau nama dari button Hello adalah "btnHello". Tujuannya adalah mengelompokkan komponen-komponen sejenis, karena pada Object Inspector nama komponen diurutkan berdasarkan alphabet. Properti **name** adalah properti internal dan digunakan untuk memberi nama pada sebuah komponen/objek. Nama ini adalah sebuah variabel yang mengacu pada komponen tersebut.

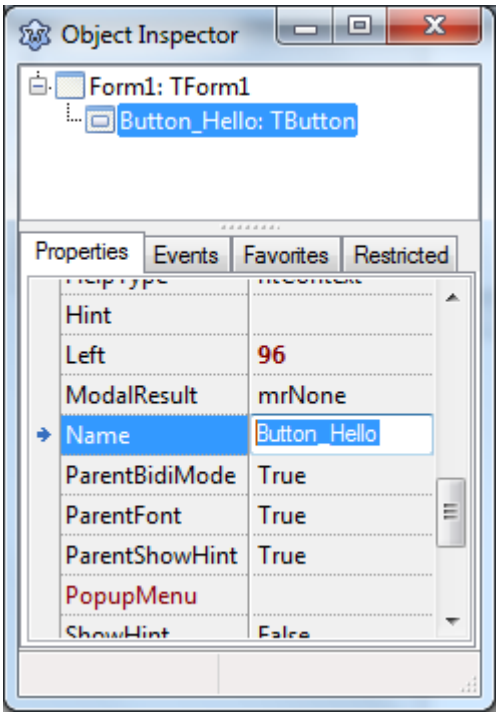
Beberapa aturan penamaan komponen atau variabel atau identifer sebagai berikut

- Diawali alpabet, berikutnya boleh angka, garis bawah.
- Tidak memakai spasi atau tanda-tanda baca atau operator kecuali garis bawah

- Boleh huruf kapital atau kecil, tidak ada perbedaan
- Tidak menggunakan kata kunci (*reserve word*) yang digunakan Lazarus
- Biasakan nama komponen diawali kelompok komponennya, misal btnHello, frmHello, rgrKelas.

Berikut contoh penamaan yang keliru menggunakan spasi

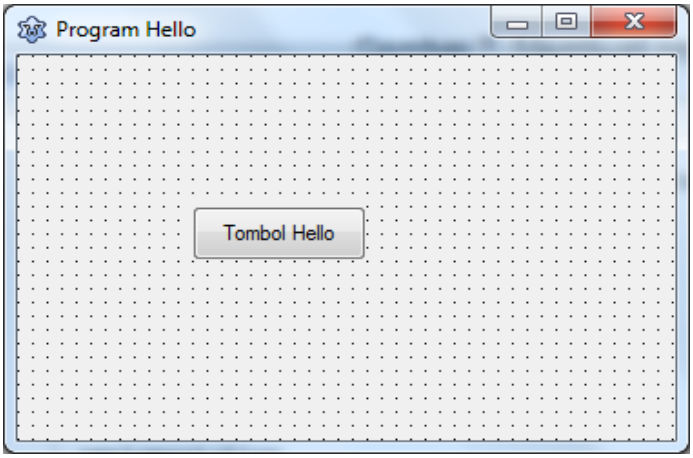
Gambar 8. Contoh Penamaan komponen yang salah (menggunakan spasi)



Gambar 9. Contoh Penamaan komponen yang benar (menggunakan garis bawah)

Kita dapat membuat nama objek yang terlihat pada Form dengan menggunakan **Caption** dalam *object inspector*. Dalam caption tidak ada aturan dalam mengisi teks seperti dalam name. Sehingga untuk **button_hello** yang tadi kita buat

misalnya kita bisa mengisinya dengan “Tombol Hello” (menggunakan spasi) dan untuk **Form1** dapat kita isis dengan “Program Hello”. Hasilnya akan terlihat di dalam Form.



Gambar 10. Membuat caption untuk Form dan button

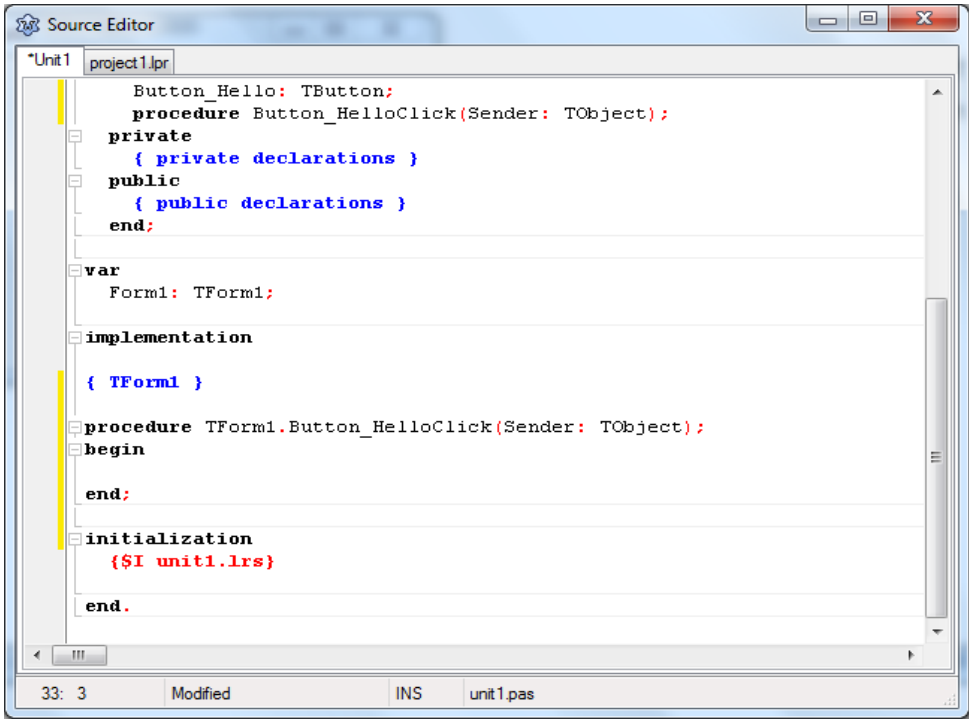
H. Membuat Method/Procedure lewat Event

Saat kita menekan tombol pada sebuah form atau komponen, Windows memberitahukan bahwa aplikasi mengirim pesan yang dibangkitkan oleh event tertentu . Lazarus akan menanggapi dengan menerima event atau panggilan tersebut. Hal ini yang dinamakan penanganan event (*event-handler method*). Lazarus mendefinisikan sejumlah event pada setiap komponennya. Daftar event ini bebeda untuk setiap komponen. Event yang paling umum pada komponen Button adalah OnClick. Artinya jika komponen Button tersebut di Klik maka akan melakukan procedure apa.

Ada beberapa teknik yang dapat dilakukan untuk menangani event misal OnClick pada komponen button :

- Klik ganda pada button tersebut, maka sebuah method/procedure btnHelloClick
- Pilih button, kemudian pilih Object Inspector’s combo box (*called the Object Selector*), pilih Tab Events, dan klik ganda pada area putih disebelah kanan event OnClick

- Pilih button, pilih Tab Events, dan masukkan nama method yang dikehendaki, misal button_HelloClick pada area putih di sebelah kanan event OnClick



Gambar 11. Kode sumber untuk button_HelloClick

Bila kita ingin menghapus procedure atau penanganan event tersebut, kita dapat menghapus pada editor Unit tersebut. Hapus blok procedure tersebut dan hapus pada bagian definisi procedure yang ada diatasnya.

Sekarang kita mengisi procedure tersebut dengan perintah untuk menampilkan pesan sebagai berikut :

```
procedure TForm1.Button_HelloClick(Sender: TObject);
begin
    MessageDlg ('Assalamu alaikum', mtInformation, [mbOK], 0);
end;
```

Perintah **MessageDlg** ini sangat sederhana, yaitu untuk menampilkan pesan. Fungsi **MessageDlg** ini mempunyai empat parameter. Untuk rincinya kita dapat

melihat Bantuan (F1).

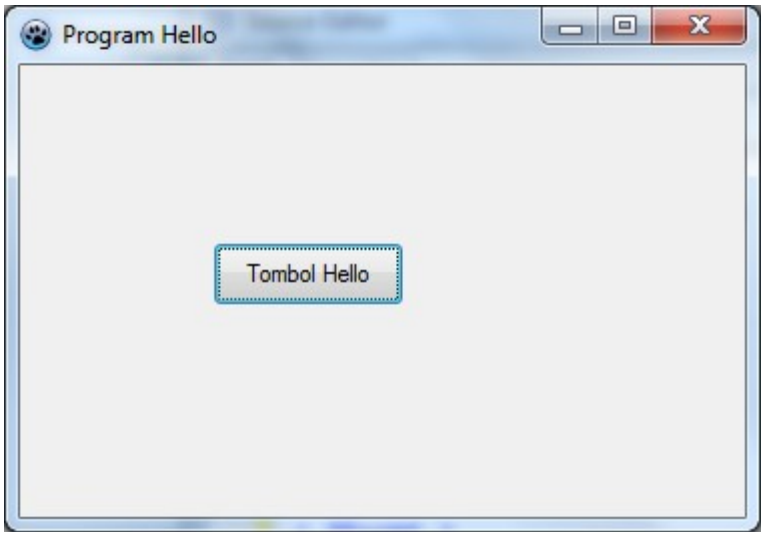
- Parameter pertama : kalimat yang akan dimunculkan (pesannya)
- Parameter kedua : tipe message box seperti mtWarning, mtError, mtInformation, atau mtConfirmation. Coba lihat perbedaannya.
- Parameter ketiga : kumpulan tombol yang akan digunakan seperti [mbYes], [mbNo], [mbOK], [mbCancel], atau [mbHelp].
- Parameter keempat : untuk help context atau nomor halaman pada Help, beri angka nol jika kita tidak mempunyai file help.

I. KoMpilasi dan Jalankan Program

Tekan tombol Run (tombol segitiga berwarna hijau di sebelah kiri atas) atau pilih menu *Run* | *Run* atau Tekan **F9**, Lazarus akan menjalankan perintah berikut ini:

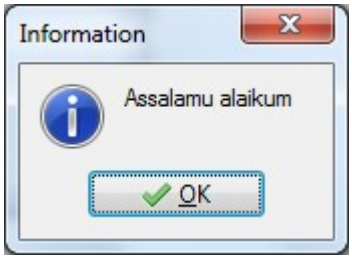
1. Kompilasi Pascal source code file yang mendefinisikan form-form yang ada (.pas, .dfm)
2. Kompilasi project file (.dpr)
3. Buat executable (.EXE) file
4. Menjalankan executable file, biasanya pada mode pencarian kesalahan (debug mode).

Setelah program executable jalam maka akan muncul jendela seperti berikut ini:



Gambar 12. Program Hello setelah dijalankan

dan apabila kita klik **Tombol Hello** akan muncul jendela seperti berikut



Gambar 13. Contoh hasil program sederhana

J. Urutan Perintah

Compiler Lazarus akan memproses semua perintah-perintah dalam kode sumber secara berurutan. Misalnya kita memiliki 3 baris perintah maka Compiler akan memroses mulai dari baris-1, baris-2 dan kemudian baris-3. Contoh program untuk menampilkan pesan sebanyak 3 kali seperti berikut :

```
MessageDlg ('Assalamu alaikum', mtInformation, [mbYes], 0);
MessageDlg ('Selamat Pagi', mtConfirmation, [mbOK], 0);
MessageDlg ('Pagi yang cerah!', mtWarning, [mbCancel], 0);
close;
```

Maka pada outputnya akan ditampilkan pesan pertama 'Assalamu alaikum' dengan logo informasi dan tombol Yes, apabila di klik tombol Yes muncul pesan kedua 'Selamat Pagi' dengan logo konfirmasi dan tombol OK dan apabila di klik tombol OK maka akan muncul pesan 'Pagi yang cerah!' dengan logo warning dan tombol Cancel. Apabila di klik Cancel program akan tertutup.

Latihan 1

Berikut latihan ke-1 untuk tiga buah aplikasi yaitu membuat tombol Hello, saat diklik tombolnya, tombol akan beraksi dengan memunculkan pesan. Ikuti langkahnya sebagai berikut :

1. Buat form frmHello1 seperti yang sudah dijelaskan.
    - Komponen yang dibutuhkan form dengan nama **frmHello1** dan button dengan nama **btnHello1**. Ubah properti *name*-nya
    - Ubah properti *caption* masing-masing komponen menjadi **Membuat program Hello1** dan **Katakan Hello**
    - Buat method dari komponen btnHello1 lewat event *OnClick,* seperti berikut **procedure** TfrmHello1.btnHello1Click(Sender: Tobject);

      begin

      MessageDlg ('Assalamu alaikum', mtInformation, [mbOK], 0);

      end;
    - Jalankan program lihat hasilnya
    - Coba ganti parameter ke-1, ke-2 dan ke-3
    - Simpan dengan nama unit *uHello1.pas* dan project *pHello1.dpr* pada direktori D:\Modul Database\
2. Buat aplikasi baru (File-New-Application) dengan form baru frmHello2 mirip form frmHello1 dengan tambahan sebagai berikut. Apa yang terjadi? (properti caption dari komponen tombol diganti saat program berjalan)
    - Komponen yang dibutuhkan form dengan nama **frmHello2** dan button dengan nama **btnHello2**. Ubah properti *name*-nya

```
procedure TfrmHello2.btnHello2Click (Sender: Tobject);
begin
    MessageDlg ('Assalamu alaikum', mtInformation, [mbOK], 0);
end;
```

K. Menu dan Perintah pada Lazarus

Ada empat cara untuk memberi perintah pada lingkungan Lazarus (Lazarus

environment):

1. Gunakan menu
2. Gunakan Short Cut (misal F9, F12 dsb)
3. Gunakan SpeedBar (atau toolbar).
4. Gunakan SpeedMenu (lokal menu yang diaktifkan dengan tombol mouse kanan).

Berikut menu utama yang ada pada Lazarus (untuk mempelajarinya gunakan Help Lazarus):

- **Menu File.** Menu ini berhubungan dengan file seperti membuat, menyimpan dan mengakhiri sebuah pekerjaan.
- **Menu Edit** .Menu ini berhubungan dengan penyuntingan apa yang dikerjakan seperti Undo , Redo, Cut, Copy, Paste atau dapat dengan tombol Ctrl+Z, Ctrl+X, Ctrl+C, Ctrl+V.
- **Menu Search**. Menu ini berhubungan dengan pencarian dan penggantian data.
- **Menu View.** Menu ini berhubungan dengan penampilan atau apa yang akan ditampilkan.
- **Menu Project.** Menu ini berhubungan dengan proyek yang sedang dibuat, misal unit yang akan ditambahkan ke proyek ini, unit apa yang akan dihapus, dsb.
- **Menu Run.** Menu ini berhubungan dengan menjalankan program, mencari kesalahan (debug), dsb.
- **Menu Component.** Menu ini berhubungan dengan komponen, misal menambah komponen baru,
- **Menu Database.** Menu ini berhubungan dengan Database, Database Form Wizard dan Database
- **Menu Tools.** Menu ini berhubungan dengan pengaturan/konfigurasi, tool-tool pembantu Lazarus.
- **Menu Help.** Menu ini berhubungan dengan informasi mengenai Lazarus, Help / bantuan

L. *Component, Property, Method, Event*

Kode yang akan dilihat pada Lazarus, serupa dengan struktur Bahasa Pascal. Lazarus adalah bahasa pemrograman berbasis objek (**Component**), artinya pendekatan pembuatan program melalui objek-objek yang ada: misalnya objek form, text, button dan sebaginya. Setiap objek akan memiliki **properti** (atribut) dan **method** yang diaktifkan/dipicu oleh **event**. Mari kita lihat penjelasan berikut.

1. Objek (COMPONENT)

Sebuah komponen merupakan sebuah objek:

- Sebuah Objek, adalah sebuah komponen dalam *Component Palette*,
- Atau sesuatu yang dibuat melalui kode-kode / bahasa pemrograman

Jadi secara umum sebuah objek adalah kelas dari kumpulan sesuatu. Komponen pasti objek, namun objek tidak selalu merupakan komponen, misal TstringList adalah sebuah objek (kumpulan karakter), dan bukan sebuah komponen.

2. PROPERTY (Atribut)

Sebuah Property tidak lain adalah sebuah nama/variabel milik sebuah objek/komponen misal Caption, Text yang dapat diubah nilai baik melalui object Inspector atau melalui program. Beberapa istilah/ nama berikut yang mirip, dan sering digunakan:

- **Procedure** adalah kumpulan perintah yang melakukan suatu proses tertentu
- **Function** adalah sama dengan procedure, tetapi proses tersebut dapat mengembalikan suatu hasil / nilai misal hasilnya = 1
- **Method** adalah procedure atau function yang tergabung pada sebuah komponen

- **Subroutine** adalah istilah umum dari semuanya (procedure/function/method) misal pada bahasa Basic.

3. METHOD (Metode)

Sebuah method adalah sebuah fungsi (*function*) yang tergabung dalam sebuah objek. Contoh ListBox (dapat berarti sebuah array of strings) yang memiliki Method (Clear) yang membuat Listbox tersebut menjadi kosong. CLEAR adalah sebuah Method pada ListBox tersebut.

```
Begin
        ListBox1.Clear; // Mengosongkan isi ListBox
        ListBox1.Items.LoadFromFile('c:\Data1.txt');
        //properti Items (bertipe string) memiliki method untuk
  LoadFromFile
end;
```

4. EVENT (Aksi)

Sebuah Event adalah sebuah aksi pengguna (User Action) misal Mouse Click, KeyPressed. Setiap Events diawali dengan kata 'On'.

Contoh :

Nama event	**Nama method**
OnClick ..	Button1Click(Sender : Tobject)
OnKeyDown ..	Button1KeyDown(Sender : Tobject)
OnMouseMove ..	Button1MouseMove(Sender : Tobject)
OnCreate	TForm1.FormCreate(Sender : Tobject)

Dan lain sebagainya

M. Cara Lazarus Bekerja

Saat kita menambahkan Components pada Form1 dan merubah nilai properti, Lazarus akan membuat kode (*pseudo code*) dalam Unit1.dfm untuk mendefinisikan apa yang kita lakukan. Untuk pembuatan program standar, kita TIDAK DIHARAPKAN MENGUBAH File Unit1.dfm. Dengan demikian dapat dikatakan bahwa kita hanya bekerja pada **Form1** dan melakukan perubahan secara Visual bukan langsung dari kode sumbernya. Hal Ini akan memudahkan kita dengan tidak perlu menghafal secara lengkap kode-kode sumber dari setiap objek tetapi cukup dengan menghafalkan nama-nama objek dan fungsi-fungsi dari properti, event dan methodnya. Model bahasa pemrograman seperti inilah yang dinamakan bahasa pemrograman Visual (*Visual Programming*)

Lazarus IDE

- Lazarus (IDE) adalah sebuah Visual Interface antara User dengan Komputer kita (yang berjalan diatas sistem operasi yang dijalankan).
- Lazarus menterjemahkan Visual Components (Buttons, Panels,..) yang ada pada Form kedalam sebuah kode-kode komputer (pseudo code dalam Unit1.dfm) yang mendefinisikan bagaimana dibentuknya form dan komponennya termasuk juga propertinya.

Mengkompilasi program (*Compiler*)

- Lazarus akan memanggil file .dpr file (file proyek kita)
- Lazarus meminta program yang ada dalam proyek tersebut dan file dpr memberikan sebagai berikut :

```
uses
Forms,
Unit1 in 'Unit1.pas' {Form1};
```

- Lazarus meminta, Apa yang dilakukan pertama kali? .dpr file memberikan sebagai berikut :

```
begin
  Application.Initialize; itializes stuff
  Application.CreateForm(Tform1, Form1);
  Application.Run;
end.
```

N. Beberapa Objek dan Method

- **Form**

  Secara default sebuah *project* Lazarus hanya memberikan sebuah Form. Tetapi dalam membuat program kita dapat bekerja dengan menambahkan beberapa form pada sebuah *project.* Beberapa event yang bisa digunakan saat berada pada sebuah Form (misalnya Form1) adalah:

  - ***Form2.Show;*** untuk membuka (*Shows*) Form2 (tetapi user diijinkan untuk dapat mengklik Form1)
  - ***Form2.ShowModal;*** Untuk membuka (*Shows*) Form2 (tetapi user tidak diijinkan untuk mengklik Form1)

- **Menampilkan Pesan (Message)**

  - ***ShowMessage*****('Ini kotak pesan');**

    Tampilan sederhana sebuah baris/teks. Pesan dituliskan didalam kurung dengan menggunakan tanda petik (' .... '). Untuk keluar dari kotak pesan (*message window*) user dapat menekan tombol OK .

  - ***MessageDlg*****('Isi kotak pesan',TMsgDlgType,[msgDlgBtn],0);**

    Mirip **ShowMessage** tapi dapat lebih dari satu tombol pilihan.

```
If MessageDlg('Please say YES or NO', mtConfirmation,
[mbYes,mbNo],0)=mrYES then
begin
        Label1.Text:='Tekan tombol YES';
end;
```

TMsgDlgType = mtWarning, mtError, mtInformation, mtConfirmation, mtCustom

msgDlgBtn = mbYes, mbNo, mbOK, mbCancel, mbAbort, mbRetry, mbIgnore, mbAll, mbHelp

Return values = mrNone, mrYes, mrNo, mrOk, mrCancel, mrAbort, mrRetry, mrIgnore, mrAll

- **Input Box (Kotak input)**

***InputBox*(' Judul Input ' , ' Apa yang akan di input ' , 'nilai default')**

```
var
InputString: string;
begin
InputString:= InputBox('Masukkan Nama', 'Please Enter your Name','');
end;
```

O. Soal-soal

1. Buatlah sebuah program untuk mengubah caption dari Form menggunakan inputbox.
2. Buatlah sebuah program dengan empat buah tombol yang mana masing-masing tombol mempunyai perintah sebagai berikut
   a) Tombol 1 bernama “Caption” memerintahkan untuk mengubah caption Tombol 2 menjadi “Informasi”, Tombol 3 menjadi “Peringatan” dan Tombol 4 menjadi “Konfirmasi”
   b) Tombol 2 bernama “Tombol 2” memerintahkan untuk menampilkan kotak dialog dengan logo Information dan button Yes
   c) Tombol 3 bernama “Tombol 3” memerintahkan untuk menampilkan kotak dialog dengan logo Warning dan button OK
   d) Tombol 4 bernama “Tombol 4” memerintahkan untuk menampilkan kotak dialog dengan logo Confirmation dan button Cancel

MODUL 2

OBJEK PASCAL DALAM LAZARUS

PENDAHULUAN

Dalam modul ini akan dipelajari mengenai objek Pascal yang digunakan dalam software Lazarus meliputi variabel, konstanta, operator matemtika dan operator string.

Modul ini terdiri atas tiga bagian yaitu pada bagian pertama Objek Pascal yang terdiri dari tipe-tipe variabel dan konstanta, konversi variabel, dan membuat komentar program.

Pada bagian kedua dijelaskan mengenai operator dasar matematika, operator akar dan pangkat, operator eksponen dan logaritma, hasil pembagian dan sisa hasil bagi, fungsi trigonometri, pembulatan bilangan, penambahan dan pengurangan bilangan integer dengan satu.

Pada bagian ketiga dijelaskan mengenai operator-operator string yang meliputi operator ord dan char, panjang string, menggabungkan string, operator untuk menyalin string, dan mencari letak string.

Bagian 1

VARIABEL DAN KONSTANTA

A. Variabel

Data yang digunakan dalam bahasa pemrograman akan disimpan didalam memori komputer. Data tersebut biasanya disimpan dalam suatu variabel atau konstanta. Sebuah variabel dalam bahasa pemrograman harus memiliki tipe data yang spesifik sehingga harus didefinisikan di awal program (dalam bagian deklarasi). Secara umum tipe data dikelompokkan menjadi 3 bagian besar yaitu:

1. **Teks** : yaitu berupa sekumpulan huruf atau angka (bisa berupa kata atau kalimat). Tipe Teks ini terdiri atas:
   a.Char (hanya terdiri dari 1 huruf/angka)
   b.String (terdiri dari beberapa huruf dan angka)
2. **Bilangan** : yaitu berupa bilangan yang terdiri atas
   a.Integer (bilangan bulat)
   b.Real (bilangan pecahan)
3. **Boolean** : Merupakan tipe variabel logika. Nilai dari variabel dari bertipe boolean adalah **True** atau **False**.

Lebih khusus di dalam Pascal (Lazarus) tipe-tipe data tersebut mempunyai beberapa varian, yang dibedakan oleh rentang datanya. Untuk penjelasannya akan diuraikan pada bagian selanjutnya

B. Konversi Tipe Data

Dalam Lazarus objek-objek visual yang disertakan yang dapat digunakan untuk input maupun output data, seperti **inputbox**, **edit**, **listbox** dan **stringgrid**, semuanya bertipe String. Oleh karena itu apabila kita ingin menggunakan suatu variabel bertipe integer atau real maka harus dikonversi dahulu menggunakan

perintah konversi. Perintah konversi dapat dilakukan didalam program yang akan dijalankan, misalnya didalam button.

Tabel 1

Konversi Data

Dari	Ke	Perintah
String	Integer	strtoint(*objek_string*)
String	Real	strtofloat(*objek_string*)
Integer	String	inttostr(*objek_integer*)
Real	String	floattostr(*objek_real*)

Contoh:

1. Misalkan kita akan menginput sebuah variabel x yang bertipe integer dari **inputbox** maka perintahnya adalah sebagai berikut

   ```
   x := strtoint(inputbox('Input Data', 'Inputkan x',''));
   ```

2. Misalkan kita akan menginputkan sebuah variabel y yang bertipe real dari objek **edit1** maka perintahnya adalah sebagai berikut

   ```
   y := strtofloat(edit1.text);
   ```

3. Sebaliknya misalkan kita akan menampilkan variabel z yang bertipe integer ke dalam label1, maka perintahnya adalah sebagai berikut

   ```
   label1.caption:=inttostr(z);
   ```

C. Integer

Integer merupakan tipe data bilangan bulat baik positif maupun negatif dalam rentang -32768 hingga 32767. Artinya apabila kita mempunyai sebuah variabel dengan tipe integer kemudian diberikan bilangan diluar rentang tersebut maka variabel tersebut tidak akan menerimanya atau apabila menerima pun akan menghasilkan suatu nilai yang salah. Dalam tipe data ini semua operasi matematika berlaku dan dapat di gunakan pada tipe real. Tabel berikut merupakan varian dari tipe data integer dengan rentang batas bawah dan batas atas serta ukuran dalam satuan bit.

Tabel 2

Varian Tipe Data Integer

TIPE	Batas Bawah	Batas Atas	Tanda	Ukuran
Shortint	–128	127	Positif/Negatif	8-bit
Smallint	–32768	32767	Positif/Negatif	16-bit
Longint	–2147483648	2147483647	Positif/Negatif	32-bit
Int64	-2^{63}	$2^{63}-1$	Positif/Negatif	64-bit
Byte	0	255	Positif	8-bit
Word	0	65535	Positif	16-bit
Longword	0	4294967295	Positif	32-bit

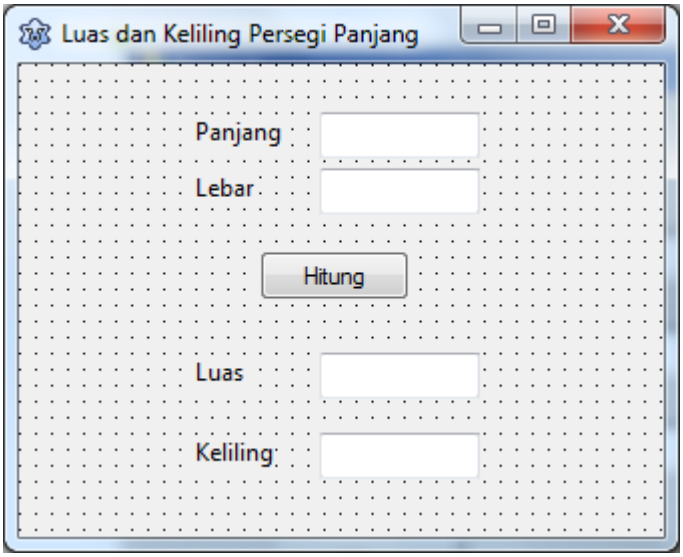
Latihan 1

Untuk lebih memahami tipe data integer kita akan coba membuat sebuah program (*proyek*) untuk menghitung Luas dan Keliling Persegi Panjang, rumus yang digunakan adalah sebagai berikut:

Luas := P * L

Keliling := 2 *(P + L)

Berikut desain form di Lazarus :



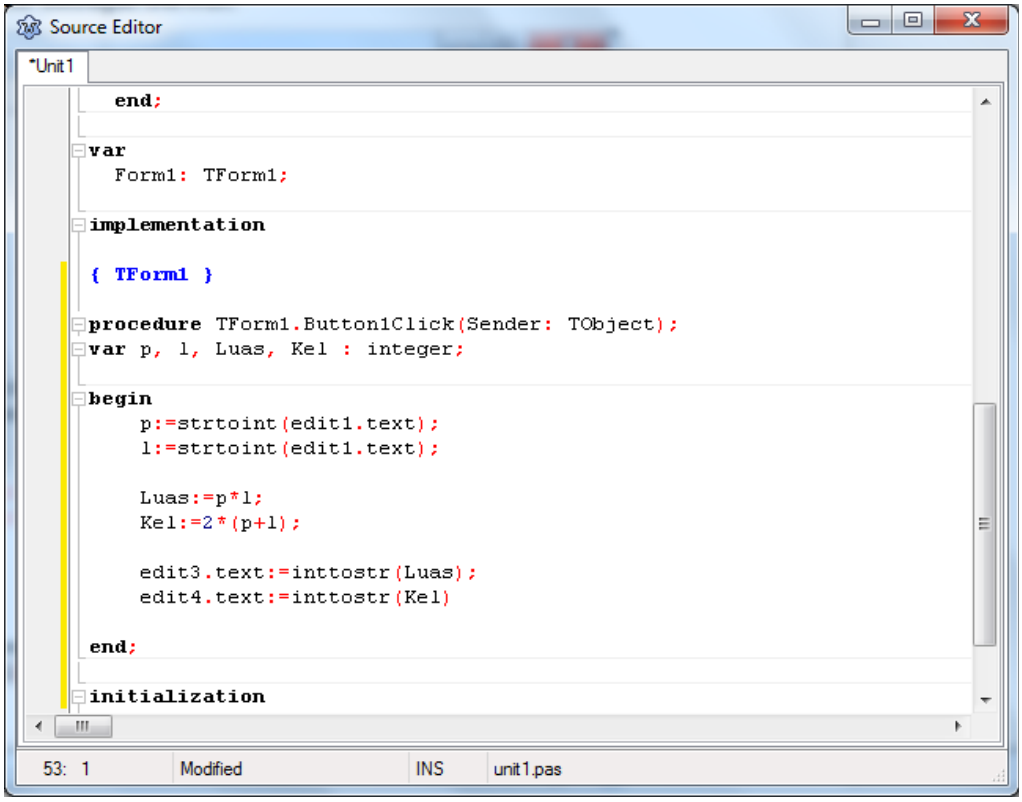
Tempatkan komponen-komponen berikut pada form kemudian ubah nilai properties-nya

Tabel 3

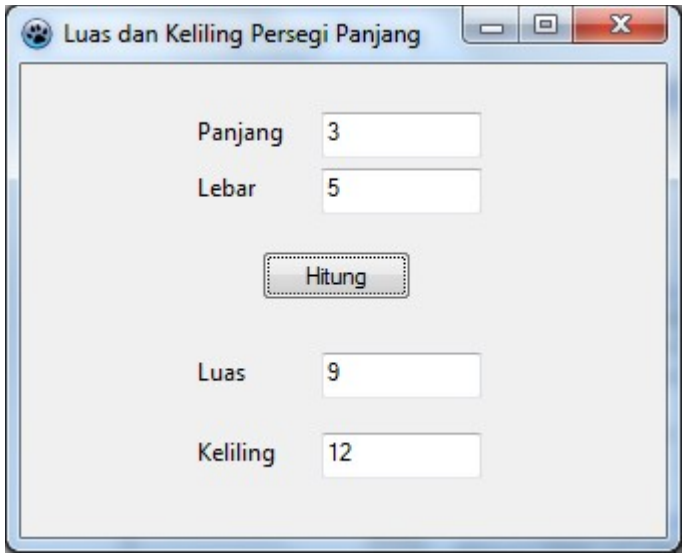
Properties Beserta Value dari Komponen

Komponen	Properties	Value
Label1	Caption	Panjang
Label2	Caption	Lebar
Label3	Caption	Luas
Label4	Caption	Keliling
Edit1	Text	
Edit2	Text	
Edit3	Text	
Edit4	Text	
Button1	Caption	Hitung

Setelah komponen diset propertiesnya seperti pada Tabel 2 Klik Dua Kali pada komponen Button dan masukkan kode sebagai berikut:



Apabila program dijalankan dengan memilih menu **run** atau menekan **F9** akan tampak hasil sebagai berikut:



D. Real

Real merupakan bilangan pecahan. Tipe bilangan ini biasanya dapat mengukur sesuatu dengan bilangan yang sangat kecil hingga bilangan yang sangat besar baik positif maupun negatif. Rentang untuk tipe real adalah 2.9 x 10^{-39} hingga 1.7 x

10^{38} untuk bilangan positif dan negatif. Tentunya tipe real ini dapat memenuhi semua operasi matematika dan dapat digabungkan dengan tipe integer. Tabel berikut merupakan varian dari tipe real dengan rentang batas bawah dan batas atas.

Tabel 4

Varian Tipe Data Real

TIPE	Batas Bawah	Batas Atas	Tanda
Real	2.9×10^{-39}	1.7×10^{38}	Positif/Negatif
Single	1.5×10^{-45}	3.4×10^{38}	Positif/Negatif
Double	5.0×10^{-324}	1.7×10^{308}	Positif/Negatif
Extended	3.4×10^{-4932}	1.1×10^{4932}	Positif/Negatif
Comp	$-2^{63}+1$	$2^{63}-1$	Positif/Negatif
Currency	–922337203685477.5808	922337203685477.5807	Positif/Negatif

Dalam Pascal metode perhitungan tipe real ada dua jenis yaitu dengan mengunakan *software* dan *co-processor*. Perhitungan menggunakan *co-processor* akan lebih cepat.

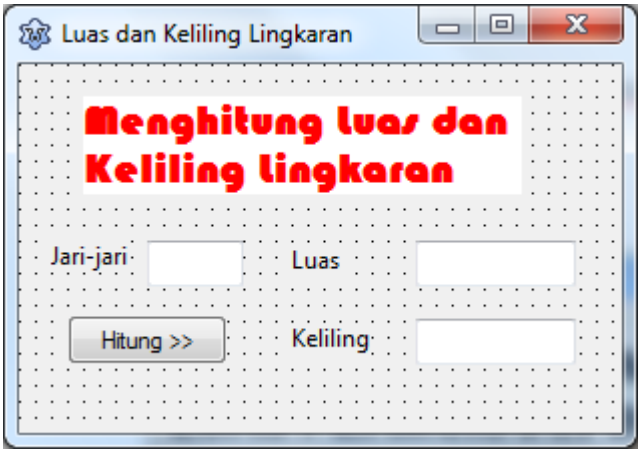
Latihan 2

Kita akan berlatih dengan tipe data real dengan membuat sebuah program (*projek*) untuk menghitung Luas dan keliling Lingkaran. Rumus yang digunakan adalah sebagai berikut:

Luas := pi* R*R

Keliling := 2*pi*R

dengan pi=3.14 adalah sebuah konstanta dalam Pascal (Lazarus). Desain Form adalah sebagai berikut:



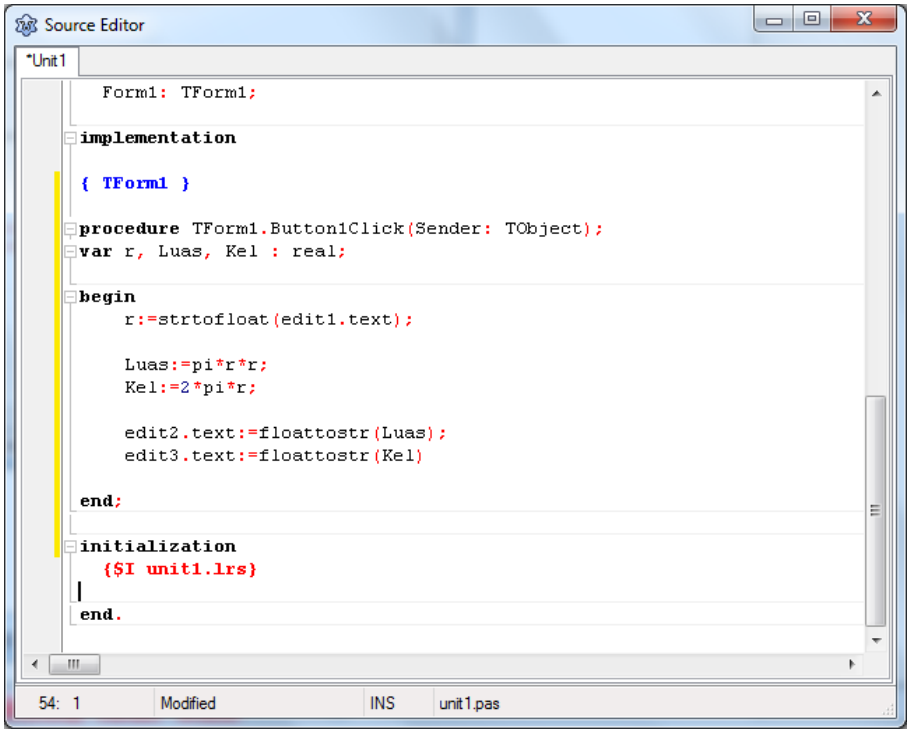
Tempatkan komponen-komponen berikut pada form kemudian ubah nilai properties-nya

Tabel 5

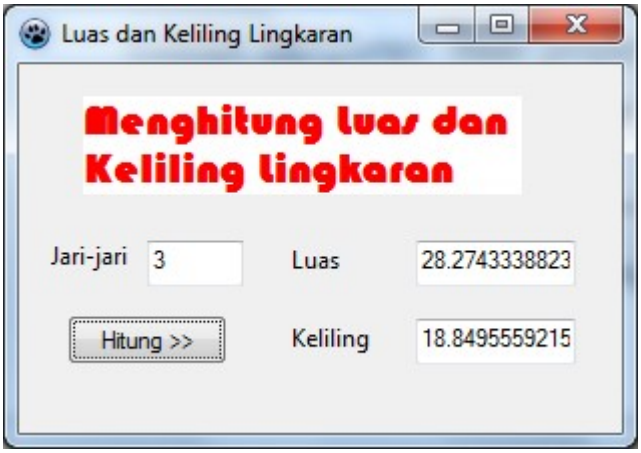
Properties Beserta Value dari Komponen

Komponen	Properties	Value
Form	Caption	Luas dan Keliling Lingkaran
Label1	Caption	Menghitung Luas dan Keliling Lingkaran
Label2	Caption	Jari-jari
Label3	Caption	Luas
Label4	Caption	Keliling
Edit1	Text	
Edit2	Text	
Edit3	Text	
Button1	Caption	Hitung >>

Setelah komponen diset propertiesnya seperti pada Tabel 5 Klik Dua Kali pada komponen Button dan masukkan kode sebagai berikut:



Apabila program dijalankan dengan memilih menu **run** atau menekan **F9** akan tampak hasil sebagai berikut:



Kita dapat mengatur properti yang lain agar tampilannya menjadi lebih menarik, misalnya mengubah bentuk dan warna font seperti pada contoh diatas.

E. String

String adalah tipe yang hnay terdiri dari kumpulan huruf, apabila diberikan sebuah angka maka pada tipe ini angka tersebut akan diperlalukan sebagai kumpulan

huruf. Pada tipe String kita tidak bisa melakukan operasi matematika. Panjang dari tipe String ini dapat didefinisikan dengan menambahkan tanda kurung siku […] setelah kata string pada bagian deklarasi. Isi dari kurung siku tersebut harus berupa bilangan bulat positif.

Tabel 6

Varian Tipe Data String

Tipe	Panjang
Char	1 character
ShortString	255 characters
AnsiString	2^31 characters
WideString	2^30 characters

Secara khusus tipe Char tidak dapat digabungkan dengan tipe string lainnya, sedangkan tipe string selain char dapat digabungkan dengan tipe string lainnya.

Latihan 3

Kita akan membuat suatu program untuk menanyakan Nomor Pelanggan, Nama Pelanggan, serta Alamat Pelanggan. Dalam hal ini semua variabel bertipe string karena tidak akan dilakukan operasi matematika terhadap variabelnya.

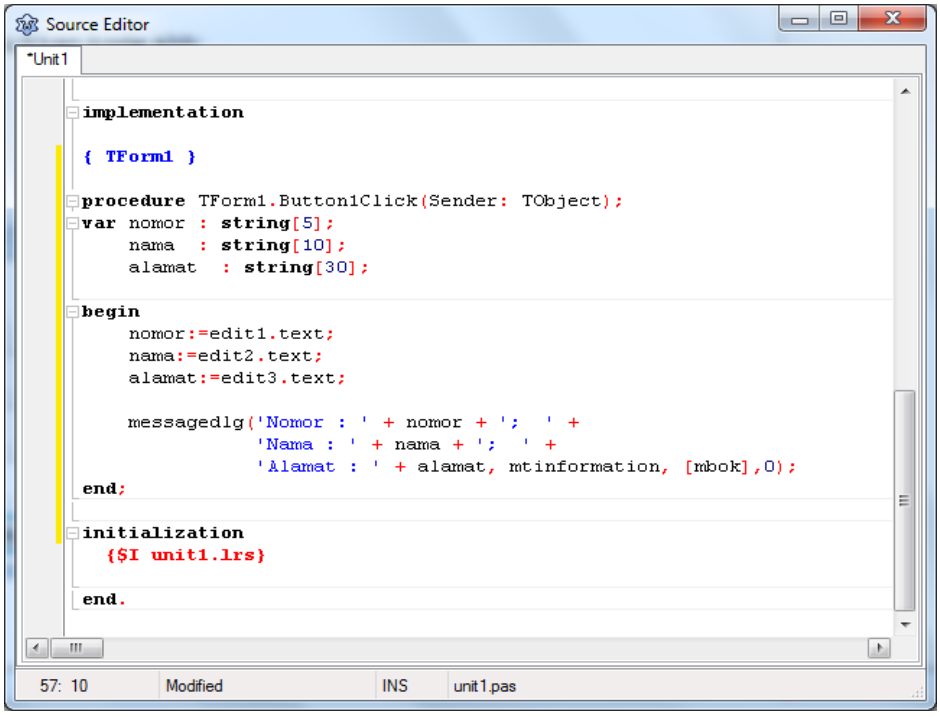
Kemudian atur properties dari setiap objek dengan mengikuti petunjuk pada Tabel 7, tentunya bisa ditambahkan dengan mengatur properti lain agar tampilannya lebih menarik.

Tabel 7

Properties Beserta Value dari Komponen

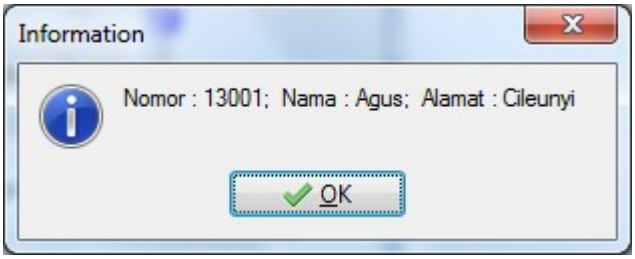
Komponen	Properties	Value
Form	Caption	Pelanggan
Label1	Caption	Data Pelanggan
Label2	Caption	Nomor
Label3	Caption	Nama
Label4	Caption	Alamat
Edit1	Text	
Edit2	Text	
Edit3	Text	
Button1	Caption	Tampilkan

Kemudian klik dua kali pada button1 dan masukkan kode sebagai berikut:



Jalankan program dengan memilih menu run atau menekan F9

Apabila kita menekan tombol tampilkan akan muncul jendela dialog sebagai berikut



F. Komentar Program

Komentar adalah bagian program yang tidak akan dibaca (dieksekusi). Tujuan dari komentar adalah memberi keterangan pada program atau bagian dari program sehingga apabila ada bagian program yang error dapat segera diketahui bagian yang mana.

Secara umum ada dua cara untuk membuat komentar program

1. Apabila komentar yang digunakan hanya satu baris maka digunakan tanda '//' di depan komentarnya
2. Apabila koementar yang digunakan lebih dari satu baris maka komentar tersebut diawali dengan '{' dan diakhiri dengan '}'.

Contoh:

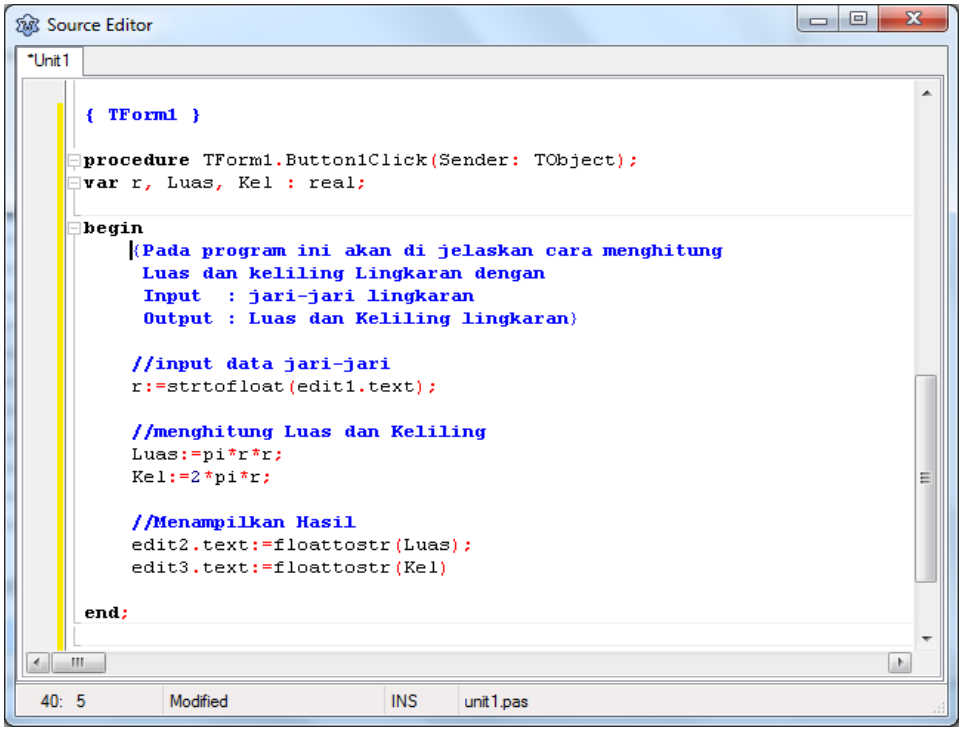
Pada program sebelumnya mengenai menghitung luas dan keliling lingkaran akan ditambahkan keterangan menggunakan kedua cara diatas. Pada awal program ditambahkan komentar mengenai program yang dibuat yaitu kalimat berikut

```
{Pada program ini akan di jelaskan cara menghitung
        Luas dan keliling Lingkaran dengan
        Input  : jari-jari lingkaran
        Output : Luas dan Keliling lingkaran}
```

Pada komentar tersebut terdiri dari lebih dari satu baris sehingga menggunakan tanda '{ ... }'. Pada setiap bagian program ditambahkan kalimat pendek berisi keterangan mengenai bagian program yaitu

```
//input data jarijari
//menghitung luas dan keliling lingkaran
//menampilkan hasil
```

Pada ketiga komentar tersebut hanya terdiri dari satu baris sehingga menggunakan tanda '//'. Perhatikan hasilnya pada gambar dibawah ini:

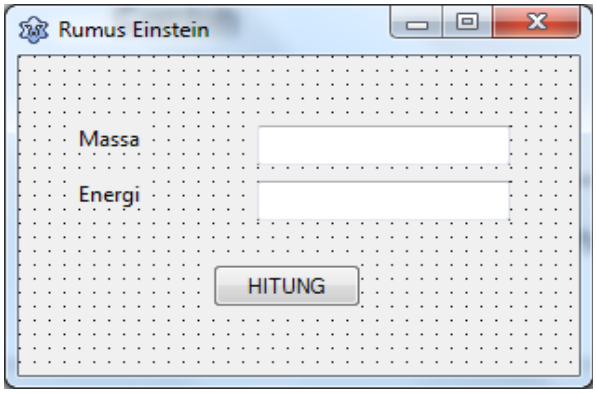


G. Konstanta

Konstanta adalah suatu harga yang nilainya tetap atau tak dapat berubah. Konstanta lebih banyak digunakan dalam perhitungan Kimia, Fisika dan Matematika. Misalnya konstanta pi, bilangan avogadro, kecepatan cahaya, kecepatan suara dan lain-lain. Beberapa konstanta telah disertakan didalam bahasa Pascal (Lazarus) sehingga tidak perlu dideklarasikan lagi, misalnya konstanta '**pi**' (π) yang bernilai 3.14159… seperti yang telah digunakan pada Latihan mengenai menghitung Luas dan Keliling Lingkaran.

Latihan 4

Pada latihan kali ini akan dibuat sebuah program untuk menghitung energi berdasarkan rumus einstein. Buatlah Form dengan desain sebagai berikut



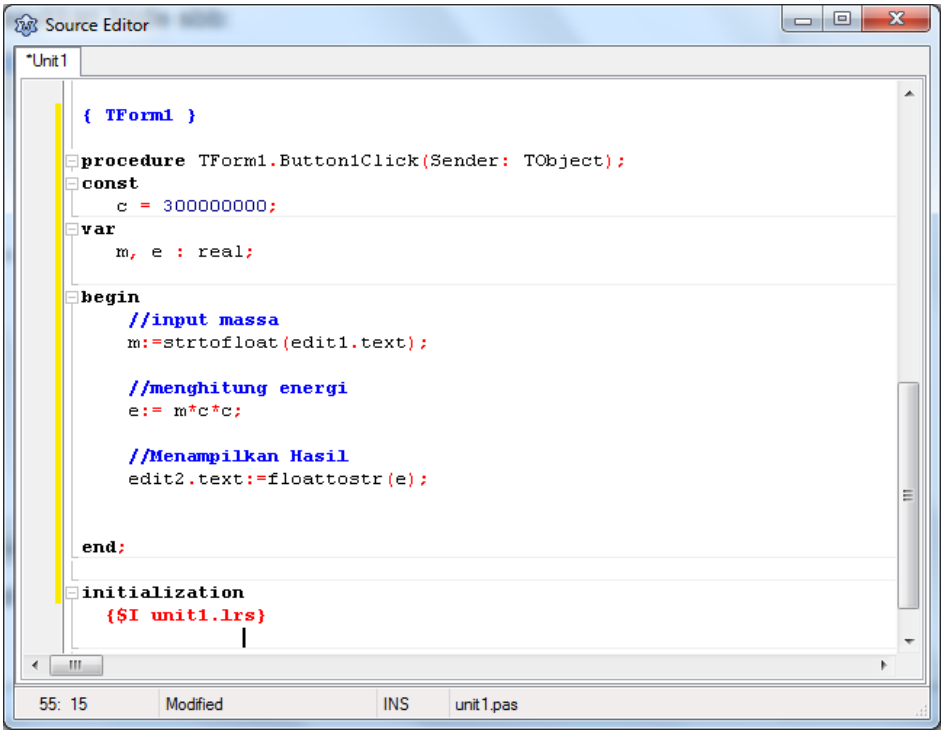
Kemudian atur propertiesnya sebagai berikut

Tabel 8

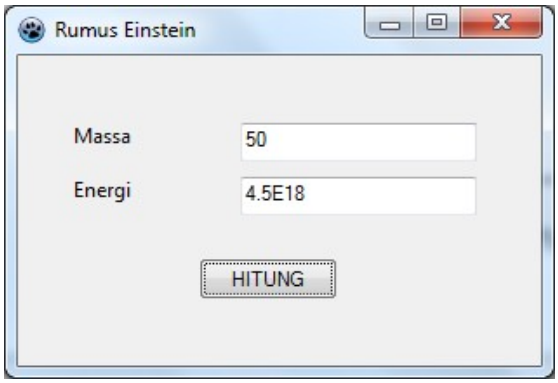
Properties Beserta Value dari Komponen

Komponen	Properties	Value
Label1	Caption	Massa
Label2	Caption	Energi
Button1	Caption	Hitung

Kemudian Klik dua kali pada button1 dan masukkan kode berikut ini:



Jalankan dengan menekan tombol running atau tekan F9 sehingga akan muncul program berikut ini



H. Soal-soal

1. Buatlah sebuah program untuk menghitung Volume dan Luas permukaan Balok dengan rumus sebagai berikut

   Volume := P * L * T

   Luas Permukaan := (2 * P * L) + (2 * P * T) + (2 * L * T)

   dengan P = Panjang, L = Lebar dan T= Tinggi

2. Buatlah sebuah program untuk menghitung Volume dan Luas permukaan Bola dengan rumus sebagai berikut

Volume := 4/3 * pi * R * R * R

Luas Permukaan := 4 * pi * R * R

dengan R = jari-jari

Bagian 2
OPERASI MATEMATIKA

A. Operator Dasar Matematika

Secara umum operator-operator dasar dalam matematika yaitu penjumlahan (+), pengurangan (-), perkalian (*) dan pembagian (/) dapat digunakan di dalam Lazarus. Beberapa hal yang perlu diperhatikan dalam melakukan operasi matematika dalam bahasa pemrograman Lazarus adalah sebagai berikut:

1. Tanda sama dengan yang digunakan adalah '**:=**'
2. Variabel yang menampung hasil operasi matematika harus berada disebelah kiri '**:=**'
3. Operasi matematika harus mengikuti tingkatan operasi matematika dan diletakkan di sebelah kanan tanda '**:=**'
4. Pada setiap persamaan operasi matematika hanya boleh ada satu tanda '**:=**'
5. Setiap kalimat harus diakhiri dengan tanda titik koma '**;**'

Tingkatan-tingkatan dalam operasi matematika adalah urutan operasi mana yang dilaksanakan terlebih dahulu. Tingkatan-tingkatan ini secara umum berlaku dalam bahasa pemrograman Pascal (Lazarus). Adapun aturan dalam tingkatan-tingkatan operasi matematika tersebut adalah:

1. Operasi dilaksanakan dari kiri ke kanan
2. Tingkatan pertama adalah semua operasi yang berada dalam tanda kurung '(...)'
3. Tingkatan kedua adalah operasi pangkat dan akar
4. Tingkatan ketiga adalah operasi perkalian dan pembagian
5. Tingkatan keempat adalah operasi penjumlahan dan perkalian

B. Fungsi SQR dan SQRT

Turbo Pascal (Lazarus) menyediakan suatu fungsi untuk menghitung kuadrat yaitu **SQR** dan akar kuadrat yaitu **SQRT**. Fungsi **SQR** dapat digunakan oleh variabel atau konstanta yang bertipe real atau integer baik positif maupun negatif, dengan tipe hasilnya sesuai dengan tipe argumen. Sementara fungsi **SQRT** dapat digunakan variabel atau konstanta yang bertipe real atau integer yang hanya bernilai positif, dan hasilnya harus bertipe real.

Contoh :

a := sqr(5) akan menghasilkan a = 25

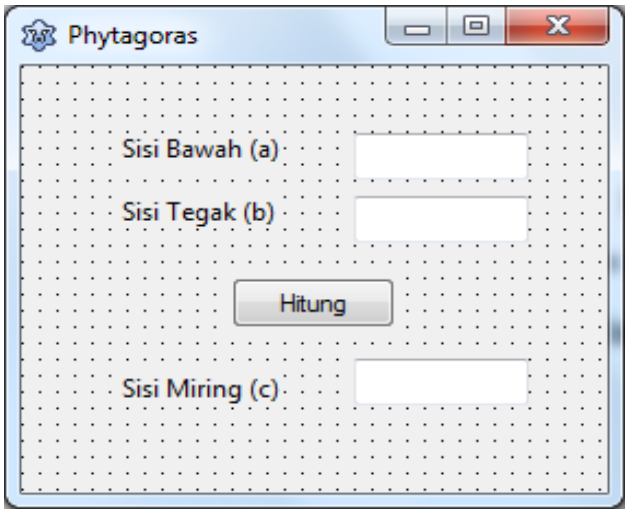
b := sqrt(16) akan menghasilkan b = 4

c := sqr(-7) akan menghasilkan c = 49

d := sqrt(-64) akan menghasilkan error karena nilai yang diakarkan negatif

Latihan 1

Kita akan membuat program untuk mencari sisi miring (c) dari rumus Phytagoras : $a^2 + b^2 = c^2$. Untuk itu akan digunakan fungsi SQR dan SQRT. Buatlah sebuah form dengan desain seperti berikut



Kemudian atur propertinya sebagai berikut

Tabel 1

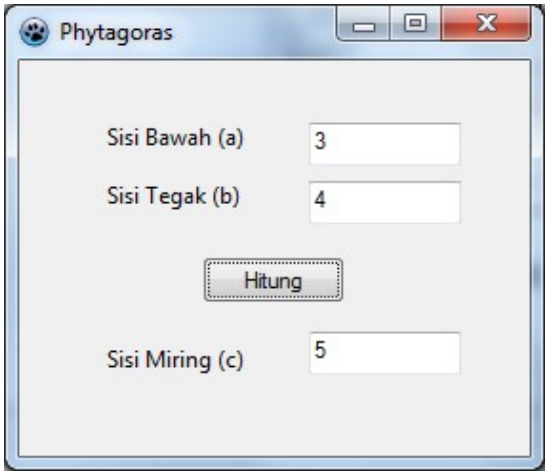
Properties Beserta Value dari Komponen

Komponen	Properties	Value
Label1	Caption	Sisi Bawah (a)
Label2	Caption	Sisi Tegak (b)
Label2	Caption	Sisi Miring (c)
Edit1	Text	
Edit1	Text	
Edit1	Text	
Button1	Caption	Hitung

Kemudian klik dua kali pada button dan masukkan kode berikut ini

```
procedure TForm1.Button1Click(Sender: TObject);
var    a,b,c : real;
begin
  //input data
     a:=strtofloat(edit1.text);
     b:=strtofloat(edit2.text);
  //perhitungan
     c:=sqrt(sqr(a)+sqr(b));
  //hasil
     edit3.text:=floattostr(c)
end;
```

Setelah di eksekusi dengan menekan tombol run atau F9 maka hasilnya adalah sebagai berikut:



C. Fungsi EXP dan LN

Eksponen yang merupakan perpangkatan dengan bilangan dasar *e* didalam Pascal (Lazarus) dituliskan sebagai **EXP**. Sedangkan Logaritma natural yaitu logaritma dengan bilangan dasar *e* di dalam bahasa pemrograman Pascal (Lazarus) dituliskan sebagai **LN**.

Contoh :

```
a := Exp (4);
b := Ln (30);
```

Fungsi **EXP** dan **LN** akan menghasilkan bilangan riil sehingga variabel hasil (a dan b dalam contoh diatas) harus bertipe real. Untuk fungsi **LN**, bilangan yang berada dalam tanda kurung harus lebih besar dari 0.

Latihan 2

Untuk berlatih penggunaan fungsi EXP dan LN kita kan membuat program untuk menghitung x^y, dengan x dan y diinputkan. Perhatikan persamaan matematika berikut ini:

$$X^y = e^{\ln(x^y)}$$

$$= e^{y \ln(x)}$$

Menggunakan persamaan tersebut maka kita dapat membuat programnya dengan desain sebagai berikut

Kemudian atur propertinya sebagai berikut

Tabel 2

Properties Beserta Value dari Komponen

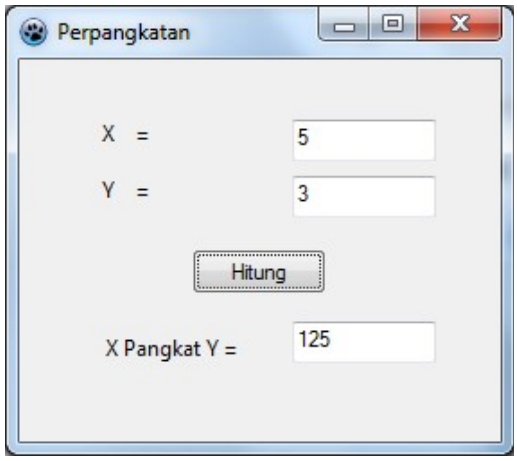
Komponen	**Properties**	**Value**
Label1	Caption	X =
Label2	Caption	Y =
Label3	Caption	X Pangkat Y =
Edit1	Text	
Edit2	Text	
Edit3	Text	
Button1	Caption	Hitung

Kemudian klik dua kali pada button dan masukkan kode berikut ini

```
procedure TForm1.Button1Click(Sender: TObject);
var    x,y, xPangkaty : real;
begin
  //input data
     x:=strtofloat(edit1.text);
     y:=strtofloat(edit2.text);
```

```
  //perhitungan
     xPangkaty:=exp(y*ln(x));
  //hasil
     edit3.text:=floattostr(xPangkaty)
end;
```

Setelah dieksekusi akan keluar hasil sebagai berikut



D. Fungsi Trigonometri

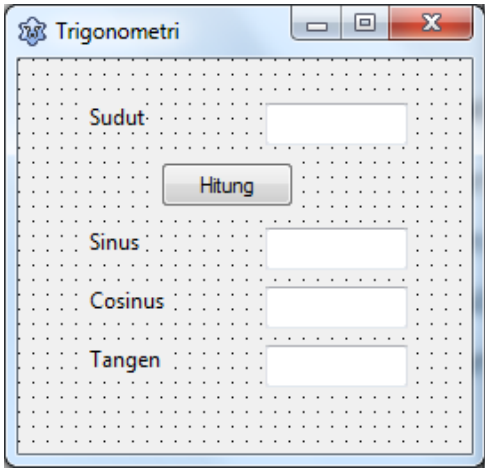
Dalam turbo pascal, fungsi trigonometri yang disediakan hanya : SIN, COS dan ARCTAN. Argumen SIN dan COS tersebut dalam satuan radian yang menghasilkan suatu bilangan real, sedangkan argumen ARCTAN adalah sebuah bilangan real yang menghasilkan sudut dalam satuan radian.

Latihan 3

Dalam latihan kali ini kita akan membuat suatu program untuk menghitung sinus, cosinus dan tangen dari suatu sudut. Adapun ketentuannya adalah sebagai berikut

Tan (a) = sin (a) * cos (a)

Untuk itu buatlah sebuah form dengan desain sebagai berikut



Kemudian atur properties sebagai berikut:

Tabel 3

Properties Beserta Value dari Komponen

Komponen	Properties	Value
Label1	Caption	Sudut
Label2	Caption	Sinus
Label3	Caption	Cosinus
Label4	Caption	Tangen
Edit1	Text	
Edit2	Text	
Edit3	Text	
Edit4	Text	
Button1	Caption	Hitung

Kemudian klik dua kali pada button dan masukkan kode berikut ini

```
procedure TForm1.Button1Click(Sender: TObject);
var   x, sinx, cosx, tanx : real;
begin
   //input sudut
   x:=strtofloat(edit1.text);
   //perhitungan
   sinx:=sin(x/180*pi);
```

```
  cosx:=cos(x/180*pi);
  tanx:=sinx/cosx;
  //output
  edit2.text:=floattostr(sinx);
  edit3.text:=floattostr(cosx);
  edit4.text:=floattostr(tanx);
end;
```

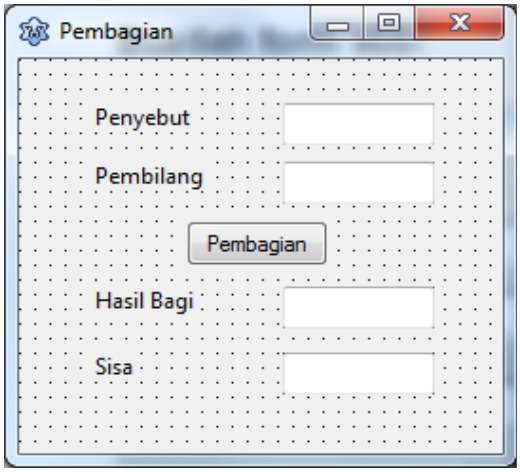
Setelah dieksekusi akan keluar hasil sebagai berikut

E. Operator MOD dan DIV

Operator Mod digunakan untuk mendapatkan sisa dari hasil pembagian dan operator Div untuk memperoleh hasil bulat pembagian.

Latihan 4

Dalam latihan kali ini kita akan membuat program untuk menghitung hasil bagi dan sisa hasil bagi. Buatlah sebuah form dengan desain seperti berikut ini:



Kemudian atur properties sebagai berikut:

Tabel 4

Properties Beserta Value dari Komponen

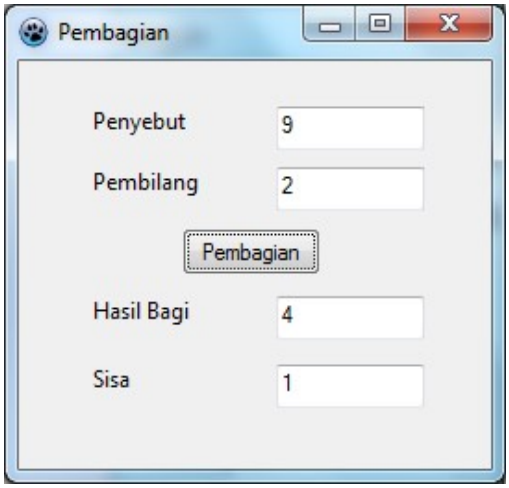
Komponen	Properties	Value
Label1	Caption	Penyebut
Label2	Caption	Pembilang
Label3	Caption	Hasil Bagi
Label4	Caption	Sisa
Edit1	Text	
Edit2	Text	
Edit3	Text	
Edit4	Text	
Button1	Caption	Pembagian

Kemudian klik dua kali pada button dan masukkan kode berikut ini

```
procedure TForm1.Button1Click(Sender: TObject);
var
  x, y, hasil, sisa : integer;
begin
   //input sudut
   x:=strtoint(edit1.text);
   y:=strtoint(edit2.text);
```

```
  //perhitungan
  hasil:=x div y;
  sisa:=x mod y;
  //output
  edit3.text:=inttostr(hasil);
  edit4.text:=inttostr(sisa);
end;
```

Setelah dieksekusi akan keluar hasil sebagai berikut

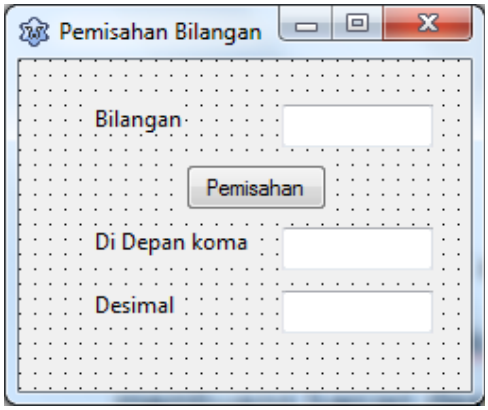


F. FungsiI TRUNC dan FRAC

Fungsi TRUNC dipakai untuk menghasilkan bilangan bulat dengan cara membuang bagian desimal suatu bilangan real. Sedangkan fungsi FRAC untuk mengambil bagian desimalnya.

Latihan 5

Untuk lebih memahami fungsi Trunc dan Frac buatlah sebuah form dengan desain sebagai berikut



Kemudian atur properties sebagai berikut:

Tabel 5

Properties Beserta Value dari Komponen

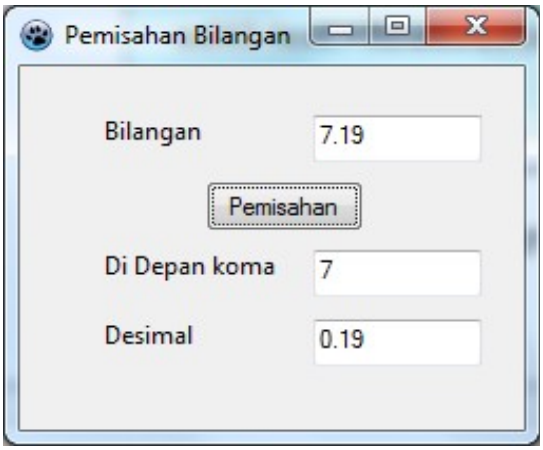
Komponen	Properties	Value
Label1	Caption	Bilangan
Label2	Caption	Di Depan Koma
Label3	Caption	Desimal
Edit1	Text	
Edit2	Text	
Edit3	Text	
Button1	Caption	Pemisahan

Kemudian klik dua kali pada button dan masukkan kode berikut ini

```
procedure TForm1.Button1Click(Sender: TObject);
var
  x,desimal : real;
  depan : integer;
begin
   //input bilangan pecahan
   x:=strtofloat(edit1.text);
   //perhitungan
   depan:=trunc(x);
   desimal:=frac(x);
```

```
  //output
  edit2.text:=inttostr(depan);
  edit3.text:=floattostr(desimal);
end;
```

Setelah dieksekusi akan keluar hasil sebagai berikut



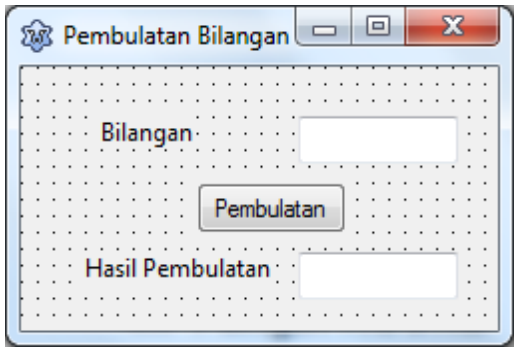
G. Fungsi ROUND

Fungsi ROUND membulatkan sebuah bilangan real ke bilangan bulat yang terdekat. Fungsi ROUND berlaku untuk aturan pembulatan bilangan sebagai berikut:

1. Jika angka di belakang koma kurang dari 5 maka angka di depan koma tetap
2. Jika angka di belakang koma lebih dari 5 maka angka di depan koma ditambahkan dengan satu
3. Jika angka di belakang koma sama dengan lima maka angka di depan koma
    - tetap apabila angka tersebut genap
    - ditambahkan dengan satu apabila angka tersebut ganjil

Latihan 6

Untuk lebih memahami fungsi round kita akan membuat suatu program sederhana dengan desain sebagai berikut



Kemudian atur properties sebagai berikut:

Tabel 5

Properties Beserta Value dari Komponen

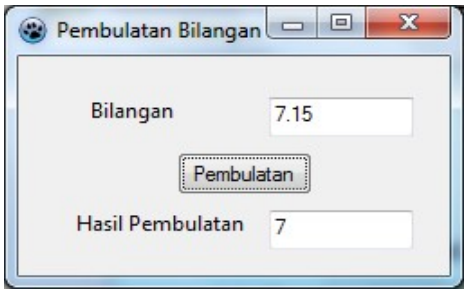
Komponen	**Properties**	**Value**
Label1	Caption	Bilangan
Label2	Caption	Hasil Pembulatan
Edit1	Text	
Edit2	Text	
Button1	Caption	Pembulatan

Kemudian klik dua kali pada button dan masukkan kode berikut ini

```
procedure TForm1.Button1Click(Sender: TObject);
var
  x : real;
  bulat : integer;
begin
   //input bilangan pecahan
   x:=strtofloat(edit1.text);
   //perhitungan
   bulat:=round(x);
```

```
  //output
  edit2.text:=inttostr(bulat);
end;
```

Setelah dieksekusi akan keluar hasil sebagai berikut



H. Penambahan dan Pengurangan dengan satu

Sebuah program biasanya ditemukan suatu pernyataan penambahan atau pengurangan bilangan integer dengan satu secara berulang-ulang. Proses seperti ini apabila digunakan operasi matematika seperti berikut ini

```
a:=a+1
b:=b-1
```

akan memakan memori yang cukup besar sehingga program akan berjalan lambat. Untuk mengatasi hal tersebut Pascal (Lazarus) menyediakan suatu fungsi untuk menambahkan atau mengurangkan bilangan dengan satu yaitu operator Inc dan Dec.

inc(a)	berarti variabel a ditambahkan dengan satu
dec(b)	berarti variabel b dikurangkan dengan satu

Penggunaan operator ini nantinya akan banyak dilakukan dalam bagian pengulangan (*looping*).

I. Soal-soal

1. Suatu perusahaan menggaji karyawannya tergantung dari Gaji Pokok, Tunjangan, serta Upah lembur. Tunjangan diberikan sebesar 20% dari gaji pokok, lembur perjam 1.5% dari gaji pokok. Buat program gaji yang diterima karyawan perusahaan tersebut.

Output yang diharapakan :

NAMA : …
NIP : …
GAJI POKOK : …
JML JAM LEMBUR : …
UPAH LEMBUR : …
TUNJANGAN : …
GAJI YANG DITERIMA : Rp …

Petunjuk : Gunakan komponen memo untuk menampilkan output.

2. Buat program untuk mencari volume tabung dan luas selimut tabung, dengan jari-jari dan tinggi diinputkan.

Rumus untuk tabung :

Luas selimut tabung = keliling alas*tinggi
Volume tabung = luas alas*tinggi

3. Jari-jari bumi sekitar 6370 km. Buat program untuk menghitung volume, luas serta keliling bumi, bila:

Keliling $= 2\pi r$
Volume $= 4\pi r^3$
Luas $= 4/3\ r^3$

Petunjuk : Gunakan fungsi SQR

4. Buatlah sebuah program untuk menentukan ukuran sampel untuk menaksir dengan input adalah proporsi (p), bound of error (B) dan N. Rumus ukuran sampel adalah sebagai berikut:

$$n = \frac{n_0}{1 + \frac{n_0 - 1}{N}} \quad \text{dengan} \quad n_0 = \left(\frac{2\sqrt{(p(1-p))}}{B} \right)^2$$

Petunjuk: ukuran sampel n selalu dibulatkan keatas.

Bagian 3

OPERASI KARAKTER DAN STRING

Secara umum karakter dan string keduanya merupakan tipe yang memuat huruf perbedaannya adalah tipe karakter (char) hanya memuat satu huruf saja sedangkan tipe string bisa memuat hingga 255 karakter. Dengan demikian kita dapat mendefinisikan karakter melalui tipe string dengan cara mengambil sebuah huruf (karakter) dari string tersebut. Tanda kurung siku '[...]' diakhir variabel string merupakan cara mengambil sebuah karakter dari variabel string dengan mengisi angka di dalam kurung siku tersebut. Misalkan kita mempunyai variabel string 'nama' yang berisi 'lazarus', apabila kita akan mengambil huruf z pada varibel 'nama' maka kita tuliskan huruf:=nama[3] sehingga variebel 'huruf' bernilai 'z'.

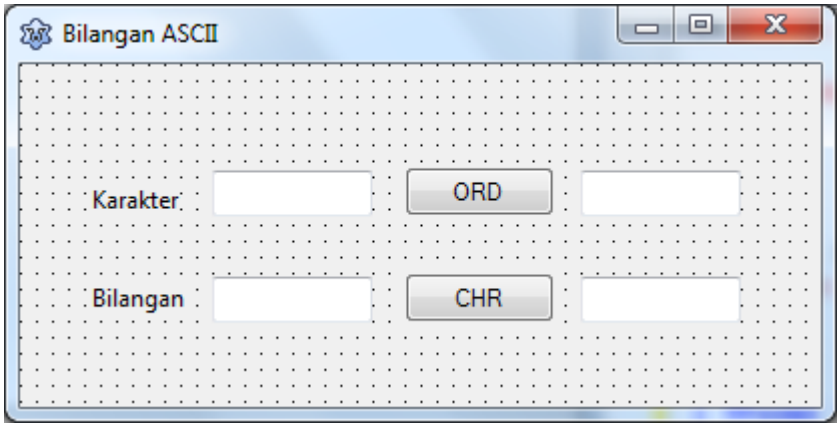
A. Fungsi ORD dan CHR

Semua huruf didalam Pascal (Lazarus) dapat dikodekan dengan suatu bilangan, bilangan itu disebut bilangan ASCII. Setiap huruf/karakter mempunyai bilangan ASCII yang berbeda-beda termasuk huruf besar dan huruf kecil, misalnya bilangan ASCII untuk huruf 'a' berbeda dengan bilangan ASCII untuk huruf 'A'.

Fungsi ORD akan menghasilkan kode suatu karakter pada tabel ASCII. Sebaliknya fungsi CHR akan menghasilkan karakter ASCII dari suatu bilangan.

Latihan 1

Kita akan membuat suatu program untuk menampilkan kode ASCII dari suatu karakter dan menampilkan suatu karakter dari sebuah bilangan ASCII. Buatlah sebuah Form dengan desain sebagai berikut:



Atur propertinya seperti pada tabel berikut ini

Tabel 1

Properties Beserta Value dari Komponen

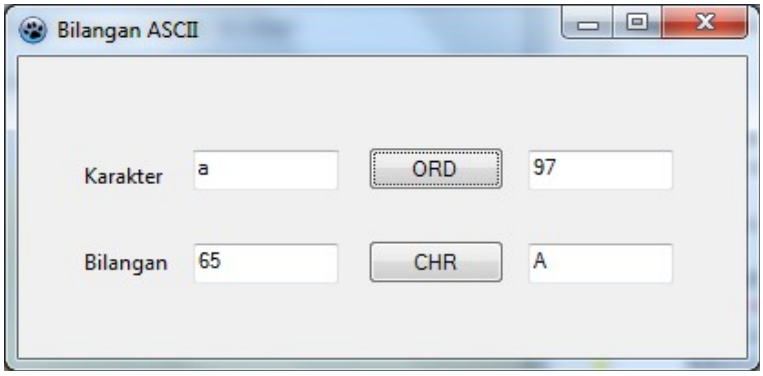
Komponen	Properties	Value
Label1	Caption	Karakter
Label2	Caption	Bilangan
Edit1	Text	
Edit2	Text	
Edit3	Text	
Edit4	Text	
Button1	Caption	ORD
Button2	Caption	CHR

Kemudian klik dua kali pada button dan masukkan kode berikut ini

```
procedure TForm1.Button1Click(Sender: TObject);
var  huruf : string;
     kar :char;
     hasil:byte;
begin
   huruf:=edit1.text;
   kar:=huruf[1];
   hasil:=ord(kar);
   edit2.text:=inttostr(hasil);
```

```
end;


procedure TForm1.Button2Click(Sender: TObject);
var  angka :byte;
    hasil : string;
begin
    angka:=strtoint(edit3.text);
    hasil:=chr(angka);
    edit4.text:=hasil;
end;
```

Setelah dieksekusi akan keluar hasil sebagai berikut



B. Fungsi UPCASE

Fungsi UPCASE akan menghasilkan huruf besar dari suatu abjad. Fungsi ini hanya akan mempunyai arti apabila datanya huruf kecil.

Contoh:

upcase('lazarus') akan memberikan hasil 'LAZARUS'

upcase('LAZarus') akan memberikan hasil 'LAZARUS'

upcase('lazaRUS') akan memberikan hasil 'LAZARUS'

upcase('LAZARUS') akan memberikan hasil 'LAZARUS'

C. Fungsi LENGTH

Panjang sebuah string dapat didefiniskan didalam Pascal (Lazarus) menggunakan fungsi LENGTH. Fungsi LENGTH digunakan untuk menghitung banyaknya karakter yang ada pada sebuah string, karakter yang dihitung termasuk tanda baca angka dan spasi. Hasil dari fungsi LENGTH bertipe integer.

Contoh:

length('bandung') akan menghasilkan nilai 7

length('Kota Bandung') akan menghasilkan nilai 12

```
kata1:='bahasa';
kata2:='pemrograman';
```

length(kata1+kata2) akan menghasilkan nilai 17.

D. Fungsi COPY

Menduplikat suatu kata atau kalimat dalam suatu string kadang dibutuhkan terutama dalam membuat suatu dokumen. Bahasa Pascal menyediakan fungsi COPY untuk menangani hal tersebut. Fungsi COPY adalah fungsi yang mengambil sebagian string dari suatu string.

Contoh:

```
kata1:='bahasa pemrograman';
hasil:=copy(kata1, 7, 11)
```

Nilai dari hasil adalah 'pemrograman'. Karena angka 7 menunjukkan karakter pertama yang akan dicopy dan angka 11 menunjukkan banyak karakter yang dicopy setelah karakter ke 7.

E, Fungsi POS

Dalam penulisan dokumen adakalanya kita ingin menemukan posisi sebuah kata dari suatu dokumen. Bahasa Pascal (Lazarus) menyediakan sebuah fungsi untuk menangani hal tersebut yaitu fungsi POS. Fungsi POS adalah fungsi untuk menentukan letak sebuah string pada string yang lain. Bila string yang dilacak tidak diketemukan, maka nilainya adalah 0 (nol).

Contoh:

```
kata1:='bahasa pemrograman';
hasil:=pos(asa, kata1)

Nilai dari hasil adalah 4 karena kata 'asa' berada di posisi karakter ke 4.
```

F. Fungsi CONCAT

Untuk menggabungkan beberapa string di dalam Pascal (Lazarus) dapat menggunakan fungsi CONCAT.

Contoh:

```
kata1:='IDE';
kata2:='Lazarus'
hasil:=concat(kata1, ' ', kata2)

Nilai dari hasil adalah 'IDE Lazarus'
```

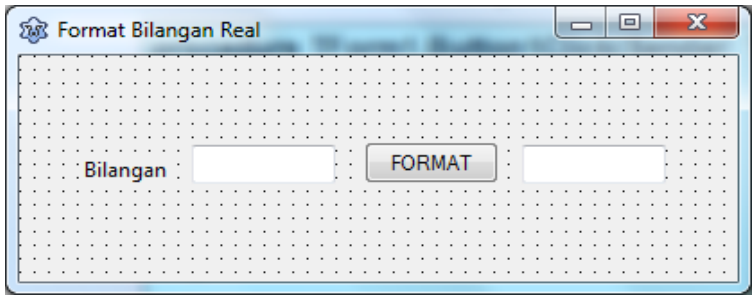
G. Prosedur FORMAT

Sebuah variabel real yang ditampilkan pada suatu output berupa text atau label mengandung banyak angka dibelakang koma sesuai dengan banyaknya hasil perhitungan. Terkadang kita tidak membutuhkan angka terlalu banyak, misalnya

saja kita hanya ingin menampilkan 2 angka dibelakang koma. Untuk itu dalam Pascal (Lazarus) digunakan prosedur FORMAT.

Latihan 2

Dalam ltihan kali ini kita akan membuat program untuk menampilkan bilangan real dengan hanya dua angka dibelakang koma. Buatlah sebuah form dengan desain sebagai berikut



kemudian atur propertinya seperti pada tabel dibawah ini

Tabel 2

Properties Beserta Value dari Komponen

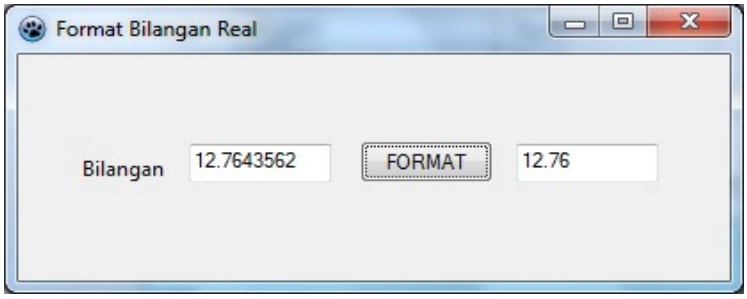
Komponen	Properties	Value
Label1	Caption	Bilangan
Edit1	Text	
Edit2	Text	
Button1	Caption	FORMAT

Kemudian klik dua kali pada button dan masukkan kode berikut ini

```
procedure TForm1.Button1Click(Sender: TObject);
var
    hasil : real;
begin
   hasil:=strtofloat(edit1.text);
   edit2.text:=format('%4.2f',[huruf]);
```

```
end;
```

Setelah dieksekusi akan keluar hasil sebagai berikut



H. Soal-soal

1. Buatlah sebuah program untuk memutar kalimat 'Bahasa Pemrograman Lazarus' menjadi 'Lazarus Bahasa Pemrograman' menggunakan operasi string.
2. Buatlah sebuah program untuk menemukan sebuah kata yang diinputkan dari kalimat berikut 'Saya sedang belajar Bahasa Pemrograman Lazarus. Bahasa Pemrograman Lazarus berbasiskan pada free Pascal yang berlisensi gratis'.
3. Salin kalimat 'Bahasa Pemrograman Lazarus' pada soal nomor 2 dan simpan hasilnya di sebuah variabel, kemudian ubah menjadi huruf besar.

MODUL 3

STRUKTUR KONDISI

PENDAHULUAN

Dalam modul ini akan dipelajari mengenai Struktur kondisi dari suatu program yang terdiri atas dua bagian yaitu struktur kondisi If..Then dan struktur kondisi Case..of.

Modul ini terdiri atas dua bagian yaitu struktur kondisi If..then yang meliputi kondisi if..then, if..then..else, kondisi lebih dari satu, pernyataan dalam kondisi dan if bertingkat atau if majemuk.

Pada bagian kedua akan dijelaskan mengenai struktur kondisi untuk lenih dari dua pilihan menggunakan Case..of yang meliputi kondisi dalam himpunan, kondisi dalam rentang dan struktur gabungan.

Bagian 1

KONDISI IF..THEN

A. Pengertian Kondisi

Sebuah kondisi adalah suatu keadaan dimana kita dihadapkan pada suatu pilihan. Untuk menyatakan kondisi biasanya terdapat dua besaran yang dibandingkan. Untuk membandingkan dua besaran dibutuhkan suatu operator relasi. Operator relasi adalah operator yang dipakai untuk membandingkan dua buah ekspresi aritmatik. Operator relasi yang digunakan di dalam Pascal (Lazarus) dapat dilihat pada tabel berikut ini

Tabel 1

OPERATOR RELASI

Operator relasi	Artinya
>	lebih besar
<	lebih kecil
>=	Lebih besar sama dengan
<=	Lebih kecil sama dengan
<>	Tidak sama dengan

B. Struktur If...Then dan If...Then....Else

Struktur IF...THEN dan IF...THE...ELSE digunakan untuk menyatakan percabangan bersyarat. Perbedaan dari IF...THEN dengan IF...THEN...ELSE adalah didalam IF...THEN tidak dibuat pernyataan apabila syarat tidak dipenuhi sedangkan dalam IF...THEN...ELSE dibuat pernyataan apabila syarat tidak dipenuhi. Sehingga dalam IF...THEN bila syaratnya dipenuhi maka pernyataan setelah THEN akan dilaksanakan, jika tidak dipenuhi tidak ada perubahan apapun sedangkan dalam IF...THEN...ELSE bila syaratnya dipenuhi maka pernyataan

setelah THEN akan dilaksanakan, jika tidak dipenuhi maka pernyataan setelah ELSE akan dilaksanakan.

Bentuk umum dari struktur IF...THEN :

```
IF < ekspresi logika >
THEN < pernyataan >;
```

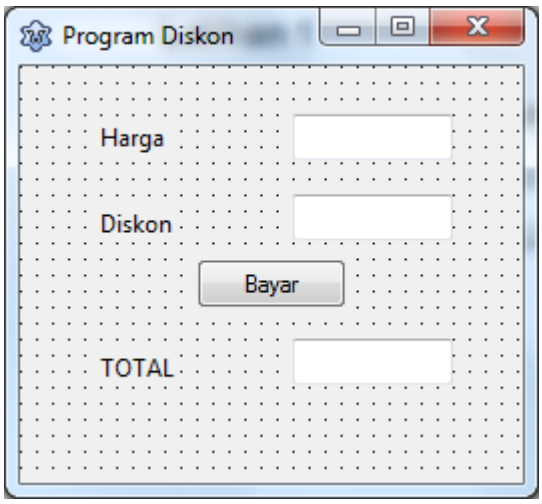
Bentuk umum dari struktur IF...THEN...ELSE :

```
IF < ekspresi logika >
THEN < pernyataan >
ELSE < pernyataan >;
```

Ekspresi logika adalah suatu pernyataan yang hanya dapat bernilai benar atau salah (TRUE or FALSE).

Latihan 1

Kita akan membuat suatu program perbelanjaan sebuah toko. Program ini program untuk memberikan potongan (diskon) sebesar 10% untuk pelanggan yang berbelanja paling sedikit dari Rp. 100000. Untuk itu buatlah sebuah Form dengan desain sebagai berikut



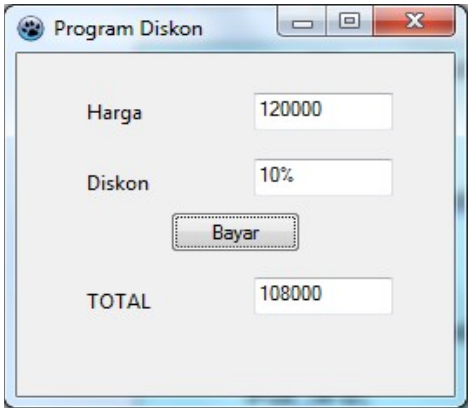
Kemudian atur properties sebagai berikut:

Komponen	Properties	Value
Button1	Caption	Bayar
Label1	Caption	Harga
Label2	Caption	Diskon
Label3	Caption	TOTAL
Edit1	Text	
Edit2	Text	0
Edit3	Text	

Klik dua kali pada button1 dan masukkan kode sebagai berikut

```
procedure TForm1.Button1Click(Sender: TObject);
var harga, pot : integer;
   total : real;
begin
   harga:=strtoint(edit1.text);
   pot:=0;
   if harga>=100000 then pot:=10;
   total:=harga-pot/100*harga;
   edit2.text:=inttostr(pot) +'%';
   edit3.text:=floattostr(total);
end;
```

Apabila program dieksekusi maka akan muncul tampilan seperti berikut



Apabila kita menggunakan struktur IF-THEN-ELSE maka kode programnya adalah sebagai berikut

```
procedure TForm1.Button1Click(Sender: TObject);
var harga, pot : integer;
   total : real;
begin
   harga:=strtoint(edit1.text);
   if harga>=100000 then pot:=10 else pot:=0;
   total:=harga-pot/100*harga;
   edit2.text:=inttostr(pot) +'%';
   edit3.text:=floattostr(total);
end;
```

Apabila dieksekusi akan menghasilkan nilai yang sama dengan sebelumnya

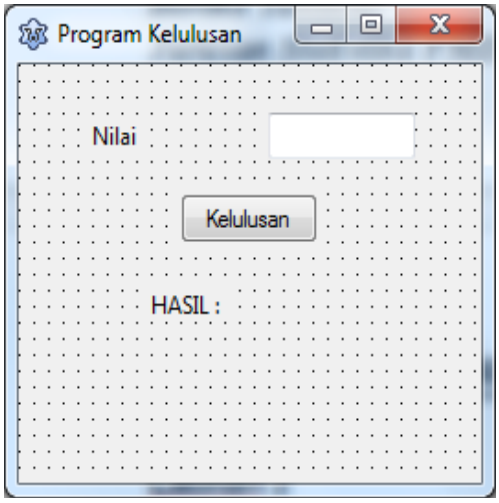
C. Koidisi Lebih dari Satu

Dalam suatu kondisi terkadang diperlukan lebih dari satu ekspresi kondisi. Untuk itu diperlukan suatu operator untuk menghubungkan kondisi yang satu dengan lainnya. Di dalam Pascal (Lazarus) operator boolean digunakan untuk menghubungkan dua buah ekspresi logika.

1. Operator AND digunakan untuk menghubungkan dua buah ekspresi logika dan **hanya** akan menghasilkan nilai benar (TRUE) apabila kedua ekspresi tersebut bernilai benar (TRUE), tetapi akan menghasilkan nilai salah (FALSE) apabila salah satu atau kedua ekspresi bernilai salah (FALSE).
2. Operator OR digunakan untuk menghubungkan dua buah ekspresi logika dan akan menghasilkan nilai benar (TRUE) apabila salah satu atau kedua ekspresi bernilai TRUE, tetapi **hanya** akan bernilai salah (FALSE) apabila kedua ekspresi bernilai salah (FALSE) .
3. Operator NOT tidak menghubungkan dua buah ekspresi logika. Operator NOT hanya memerlukan sebuah ekspresi logika. Operator NOT dipakai untuk membalikkan nilai ekspresi logika. Operator NOT bernilai TRUE jika ekspresi logika bernilai FALSE begitu pula sebaliknya.
4. Operator IN bekerja pada sebuah himpunan. Operator IN akan memeriksa apakah suatu nilai merupakan anggota dari suatu himpunan. Hasilnya TRUE bila nilai tersebut merupakan anggota himpunan.

Latihan 2

Pada Latihan berikut kita akan membuat program mengenai operator AND.
Buatlah form seperti gambar berikut ini



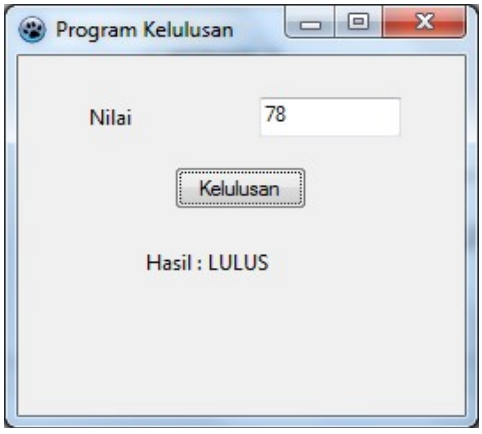
Kemudian atur properties sebagai berikut:

Komponen	Properties	Value
Button1	Caption	Kelulusan
Label1	Caption	Nilai
Label2	Caption	Hasil :
Edit1	Text	

Kemudian klik dua kali pada button dan masukkan kode berikut

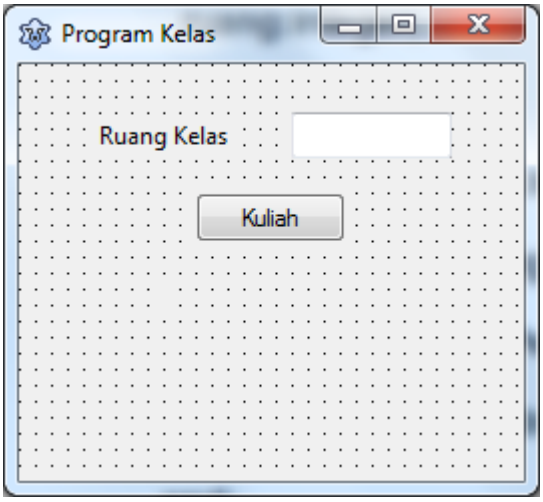
```
procedure TForm1.Button1Click(Sender: TObject);
var nilai : real;
begin
   nilai:=strtoint(edit1.text);
   if (nilai>=70) and (nilai<=100)  then
        label3.caption:='Hasil : LULUS'
   else   label3.caption:='Hasil : TIDAK LULUS';
end;
```

Apabila program dieksekusi maka akan muncul tampilan seperti berikut



Latihan 3

Pada Latihan berikut kita akan membuat program mengenai operator OR. Buatlah form seperti gambar berikut ini



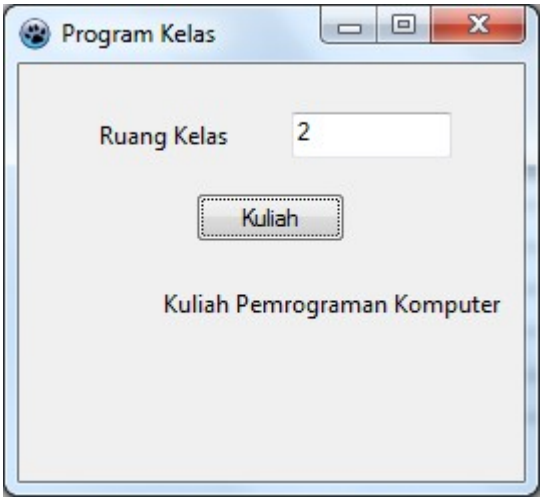
Kemudian atur propertiesnya sebagai berikut :

Komponen	Properties	Value
Button1	Caption	Kuliah
Label1	Caption	Ruang Kelas
Label2	Caption	
Edit1	Text	0

Kemudian Klik dua kali pada button1 dan masukkan kode berikut

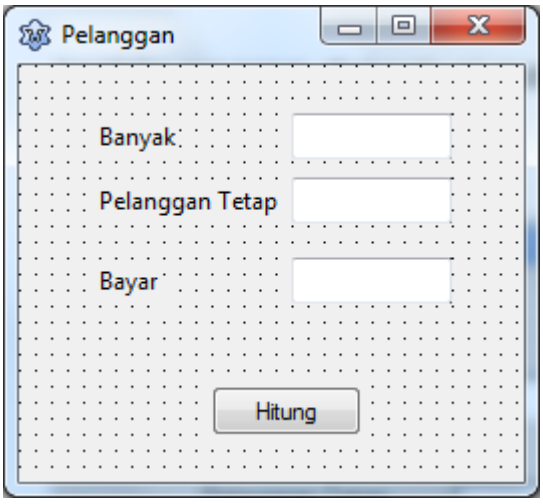
```
procedure TForm1.Button1Click(Sender: TObject);
var ruang : integer;
begin
   ruang:=strtoint(edit1.text);
   if (ruang=2) or (ruang=3)  then
         label3.caption:='Kuliah Pemrograman Komputer'
   else   label3.caption:='Kuliah Komputasi Statistika';
end;
```

Apabila program dieksekusi maka akan muncul tampilan berikut ini



Latihan 4

Pada Latihan berikut kita akan membuat program mengenai operator IN. Pada program ini pelanggan tetap mendapatkan potongan harga. Buatlah form seperti gambar berikut ini



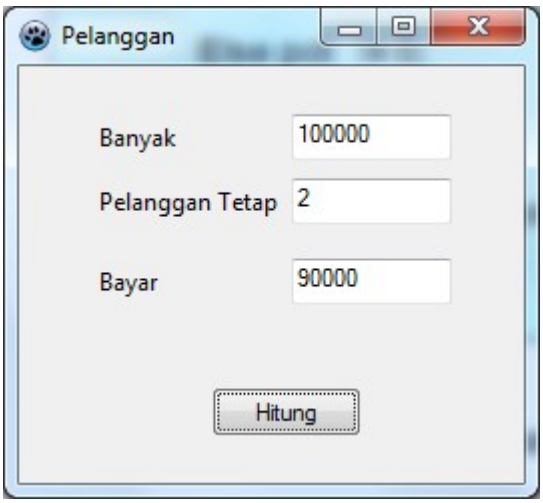
Kemudian atur propertiesnya sebagai berikut :

Komponen	Properties	Value
Button1	Caption	Bayar
Label1	Caption	Banyak
Label2	Caption	Pelanggan Tetap
Label3	Caption	Bayar
Edit1	Text	
Edit2	Text	
Edit3	Text	

Kemudian Klik dua kali pada button1 dan masukkan kode berikut

```
procedure TForm1.Button1Click(Sender: TObject);
var
    tetap:integer;
    banyak,bayar,pot:real;
begin
    banyak:=strtofloat(edit1.text);
    tetap:=strtoint(edit2.text);
    If tetap in [1,2] Then pot := 0.1 * banyak
            Else pot  := 0;
    Bayar  := banyak - pot ;
    edit3.Text:=floattostr(bayar);
end;
```

Apabila program dieksekusi maka akan muncul tampilan berikut ini



D. Struktur IF Bertingkat

Suatu struktur IF...THEN...ELSE memungkinkan adanya struktur IF...THEN...ELSE didalmnya. Hal ini dilakukan apabila sebuah kondisi memerlukan suatu pernyataan kondisi lain yang berada didalam pernyataan. Secara umum baik bagian THEN maupun bagian ELSE bisa mempunyai struktur

kondisi lain. Struktur seperti ini disebut sebagai struktur IF MAJEMUK atau IF BERTINGKAT Untuk lebih jelasnya perhatikan bagan berikut :

```
IF<ekspresi logika>
THEN
        Begin
                <pernyataan>
                IF <ekspresi logika>
                THEN<pernyataan>;
                ELSE<pernyataan>;
        End;
ELSE
        Begin
                <pernyataan>
                IF <ekspresi logika>
                THEN <pernyataan>;
        End;
```

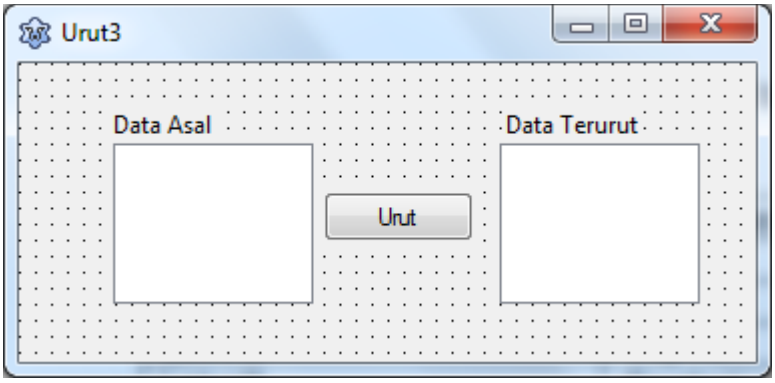
E. Blok Statement

Blok Stament adalah kesatuan dari beberapa pernyataan. Biasanya Blok Statement digunakan apabila ada pernyataan yang lebih dari satu baris. Struktur IF...THEN atau IF...THEN...ELSE memerlukan blok jika bagian pernyataan setelah THEN atau bagian pernyataan setelah ELSE ada lebih dari suatu pernyataan. Sebuah blok dibatasi oleh kata simpan *begin* dan *end.* Pernyaan Blok dapat diperhatikan pada bagan dalam struktur IF MAJEMUK diatas.

Latihan 5

Dalam latihan kali ini kita akan membuat program untuk mengurutkan *n* buah data menggunakan IF Bertingkat. Buatlah sebuah Form dengan desain sebagai

berikut



Kemudian atur propertiesnya sebagai berikut :

Komponen	Properties	Value
Button1	Caption	Urut
Label1	Caption	Data Asal
Label2	Caption	Data Terurut
Listbox1	Caption	
Listbox2	Caption	

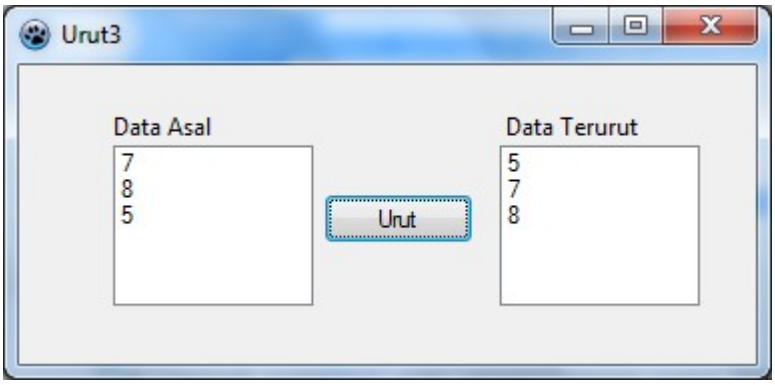
Kemudian Klik dua kali pada button1 dan masukkan kode berikut

```
procedure TForm1.Button1Click(Sender: TObject);
var x1,x2,x3,y1,y2,y3:real;
begin
  //input data
  listbox1.Items.Add(inputbox('Input Data', 'X1 =', ''));
  listbox1.Items.Add(inputbox('Input Data', 'X2 =', ''));
  listbox1.Items.Add(inputbox('Input Data', 'X3 =', ''));
  x1:=strtofloat(listbox1.Items[0]);
  x2:=strtofloat(listbox1.Items[1]);
  x3:=strtofloat(listbox1.Items[2]);
  //proses pengurutan
  if x1<=x2 then
```

```
  if x1<=x3 then
        if x2<=x3 then
        begin
         y1:=x1;y2:=x2;y3:=x3;
        end
      else
        begin
         y1:=x1;y2:=x3;y3:=x2;
        end
  else
    begin
      y1:=x3;y2:=x1;y3:=x2;
    end
else
  if x1>x3 then
        if x2>x3 then
      begin
        y1:=x3;y2:=x2;y3:=x1;
      end
    else
      begin
        y1:=x2;y2:=x3;y3:=x1;
      end
  else
    begin
      y1:=x2;y2:=x1;y3:=x3;
    end;
//output data
```

```
  listbox2.Items.Add(floattostr(y1));
  listbox2.Items.Add(floattostr(y2));
  listbox2.Items.Add(floattostr(y3));
end;
```

Apabila program dieksekusi maka akan muncul tampilan berikut ini



Bagian 2
KONDISI CASE..OF

A. Struktur Case...Of

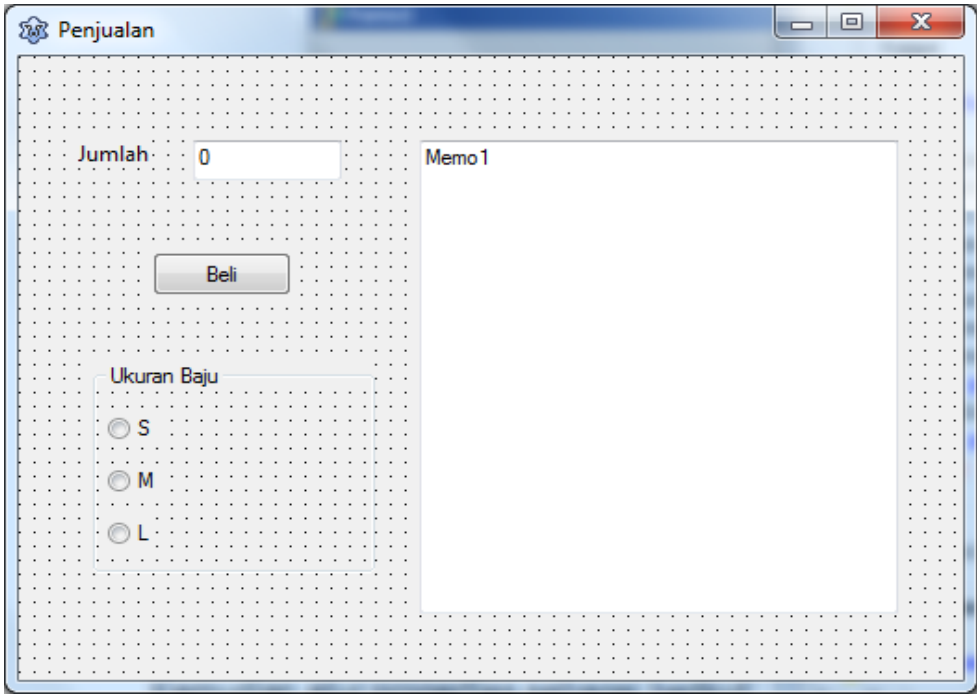
Struktur kondisi terkadang membutuhkan pilihan lebih dari dua sehingga penggunaan IF...THEN...ELSE akan menemui kendala. Pascal (Lazarus) menyediakan struktur CASE...OF yang dapat digunakan untuk memilih dengan kemungkinan lebih dari dua. Secara umum bentuk struktur CASE...OF adalah sebagai berikut:

```
Case <variable> of
        <konstanta 1> :  <pernyataan>
        <konstanta 2> :  <pernyataan>
        <konstanta 3> :  <pernyataan>
          .
          .
        Else <pernyataan>
End;
```

Dalam struktur CASE...OF Variabel dan konstanta yang dapat digunakan hanyalah yang mempunyai tipe integer atau char.

Latihan 5

Untuk lebih memahami struktur CASE...OF kita akan membuat program untuk menghitung jumlah pembelian baju dengan tiga ukuran yaitu S, M dan L. Untuk itu buatlah sebuah Form dengan desain sebagai berikut:



Kemudian atur properties-nya sebagai berikut:

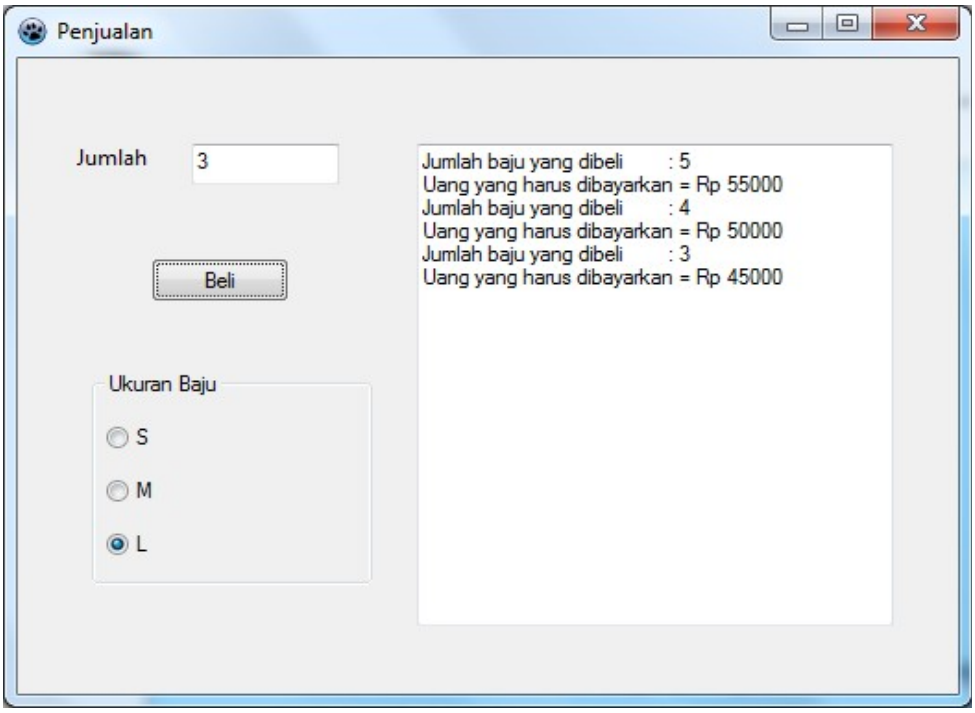
Komponen	Properties	Value
Button1	Caption	Beli
Label1	Caption	Jumlah
Edit1	Text	0
RadioGroup1	Caption	Ukuran Baju
	Items	S,M,L
	Name	Baju
Memo1	Lines	

Kemudian klik dua kali pada button1 dan masukkan kode berikut

```
procedure TForm1.Button2Click(Sender: TObject);
Var
        banyak : integer;
        harga, jumlah : real;
begin
  banyak:=strtoint(edit1.Text);
        Case baju.ItemIndex  of
```

```
                0:harga:= 11000;
                1:harga:= 12500;
                2:harga:= 15000;
        End;
        Jumlah :=banyak*harga;
 memo1.Lines.Add('Jumlah baju yang dibeli      : '+inttostr(banyak));
      memo1.Lines.Add ('Uang yang harus dibayarkan = Rp
'+floattostr(jumlah));
end;
```

Apabila program dieksekusi maka akan muncul tampilan berikut ini



B. Case...Of dengan Beberapa Konstanta dan Rentang Konstanta

Sebuah CASE..OF diperbolehkan mempunyai lebih dari satu konstanta. Selain itu konstanta tersebut juga bisa diganti dengan rentang (*range*) tertentu. Rentang

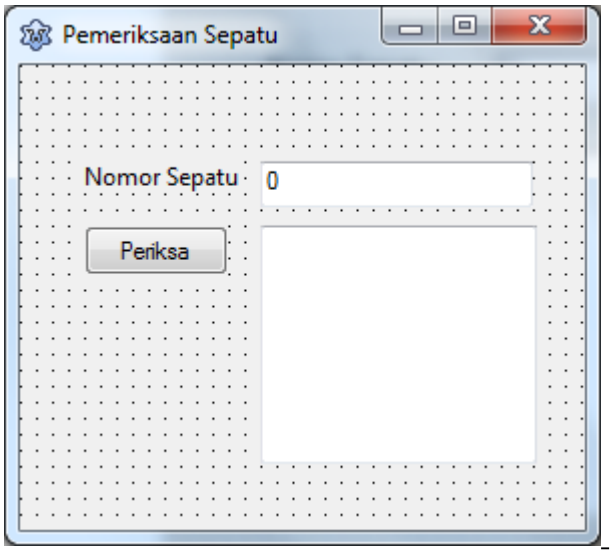
didalam Pascal didefinisikan menggunakan tanda '..'. Selain itu konstanta yang digunakan bisa merupakan kombinasi dari dua cara tersebut.

Contoh

```
Case nomor of
        1..5,8 : harga : 15000;
        6..7,9: harga : 20000;
        10..12: harga : 25000;
End.
```

Latihan 6

Dalam latihan kali ini kita akan membuat program untuk memeriksa nomor sepatu. Untuk itu buatlah sebuah Form dengan desain sebagi berikut:



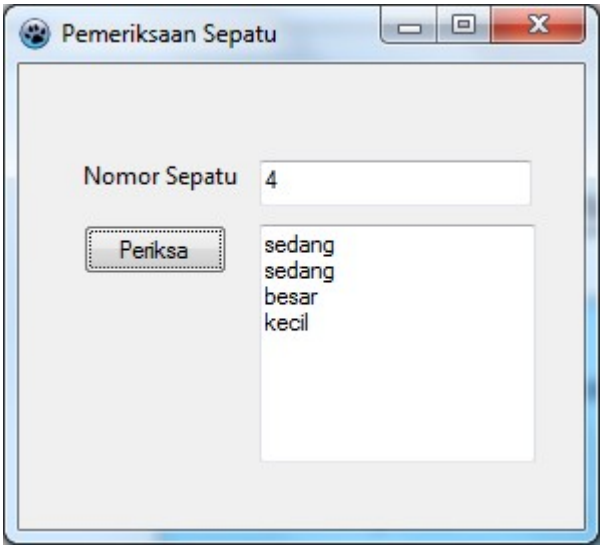
Kemudian atur properties-nya sebagai berikut:

Komponen	**Properties**	**Value**
Button1	Caption	Periksa
Label1	Caption	Nomor Sepatu
Edit1	Text	0
Memo1	Lines	

Klik dua kali pada button1 dan masukkan kode berikut

```
Procedure TForm1.Button3Click(Sender: TObjet);
var
 nomor:integer;
 sepatu:string;
begin
 nomor:=strtoint(edit1.Text);
 Case nomor of
    1..5:Sepatu:='kecil';
    6..10:sepatu:='sedang';
   11..15:sepatu:='besar';
   16..24:sepatu:='raksasa';
  end;
  memo1.Lines.Add(sepatu);
end;
```

Kemudian apabila program di eksekusi makan akan muncul tampilan berikut



C. Struktur Campuran

Setiap struktur kondisi baik IF..THEN..ELSE maupun Struktur CASE..OF mempunyai kelebihan dan kekurangannya masing-masing. Sehingga sangat dimungkinkan dalam membuat program untuk menggabungkan kedua struktur tersebut. Sebuah struktur CASE...OF boleh mempunyai struktur CASE..OF lain didalamnya atau struktur lainnya, misalnya IF-THEN-ELSE. Bentuk umumnya dapat dilihat dalam bagan berikut ini:

```
Case .. of
        .. : begin
                …
             end;
        ..: begin
                …
                case … of
                        … : …
                end;
                end;
             .. : if …
                        then …
                        else …
           end;
```

D. Soal-soal

1. Sebuah supermarket memberikan potongan sebesar 12.5% untuk jumlah pembelian dari kelebihannya dari Rp 83000. kitaikan besar pembelian adalah Rp 100000,maka yang mendapat potongan adalah yang Rp 17000. Buat programnya!
2. Sebuah Rental CAR mempunyai peraturan sebagai berikut, satu jam pertama sewa mobil adalah Rp 25000, jam berikutnya biaya sewa adalah Rp 15000. Apabila kita menyewa mobil selama 3 jam,maka

perhitungannya adalah(1 * 25000) + (2 * 15000). Buatlah program untuk menghitung lama penyewaan dan berapa besar si pelanggan harus membayar !

3. Sebuah lembaga kursus komputer mempunyai 3 kriteria dalam pemungutan biaya ujian ujian lokal dan ujian negara/SKS tiap kriteria adalah sebagai berikut:

KRITERIA	UJIAN LOKAL	UJIAN NEGARA/SKS
Tahun ke 1	Rp 12500	Rp 6000
Tahun ke 2	Rp 15000	Rp 7500
Tahun ke 3	Rp 22500	Rp 10000

Ujian negara diberikan kepada mahasiswa yang telah menempuh jumlah sks lebih dari12 SKS. Buat program untuk menghitung jumlah biaya yang harus disediakan oleh seorang mahasiswa.

Input : Nama mahasiswa, Kriteria, dan jumlah SKS.

Output : Jumlah uang ujian

Petunjuk: Gunakan struktur CASE dengan BLOK(begin ... end).

4. Penemuan indeks prestasi mahasiswa adalah sebagai berikut :

Nilai >= 90 : A

75<= nilai < 90 : B

65<= nilai < 75 : C

60<= nilai < 65 : D

nilai < 60 : E

Buat programnya untuk mengetahui indeks prestasi suatu mata kuliah. Buat dalam 3 cara penyelesaian yaitu:

a. menggunakan struktur IF-THEN dengan operator AND dan IN

b. menggunakan struktur IF-THEN-ELSE majemuk

c. menggunakan struktur CASE

5. Di beberapa toko komputer menjual aneka disket pemrograman original, dengan harga yang beraneka ragam pula yaitu:

Program	Toko_A	Toko_B	Toko_C
Clipper	450000	435000	462500
Pascal	225000	187000	197500
Fortran	315000	300000	305500
Assembler	220000	200000	202500

Buat programnya untuk menghitung jumlah yang harus dibayar oleh seorang pembeli disket. Input : Jenis disket (1-4), Toko(A/B/C), banyak yang dibeli. Output : Jumlah yang harus dibayar.

Petunjuk: Gunakan struktur CASE di dalam CASE.

MODUL 4

PENGULANGAN

PENDAHULUAN

Dalam modul ini akan dipelajari mengenai jenis-jenis pengulangan dalam bahasa pemrograman meliputi pengulangan For dan pengulangan dengan kondisi yaitu while..do dan repeat..until.

Modul ini terdiri atas dua bagian yaitu bagian pertama mengenai pengulangan menggunakan for..do. Sedangkan pada bagian kedua dijelaskan mengenai pengulangan dengan kondisi yaitu struktur while..do dan repeat..until.

Struktur pengulangan atau Looping merupakan struktur yang sangat penting dalam bahsa pemrograman. Karena dengan menggunakan struktur pengulangan kita dapat menghemat banyak memori dalam pengerjaan program kita. Oleh karena itu Lazarus menyediakan tiga jenis pengulangan yaitu **For...Do**, **While...Do** dan **Repeat...Until**. Ketiga jenis pengulangan ini dapat digunakan dalam kondisi pengulangan yang berbeda-beda.

Bagian 1

PENGULANGAN FOR..DO

A. Pengulangan FOR .. DO

Struktur Pengulangan FOR...DO merupakan pengulangan dengan jumlah yang ditentukan. Pengulangan ini bergantung dari suatu variabel yang bertipe integer. Dalam pengulangan FOR...DO harus ditentukan nilai awal dan nilai akhir dari proses pengulangannya. Setiap pengulangan FOR...DO variabel tersebut akan bertambah atau berkurang dengan satu. Secara umum bentuk pengulangan FOR-DO ada 2 yaitu

1. Pengulangan dengan variabel bertambah dengan satu

   FOR <variable pengontrol> : = <batas bawah> **to** <batas atas> **do** <pernyataan>

2. Pengulangan dengan variabel berkurang dengan satu

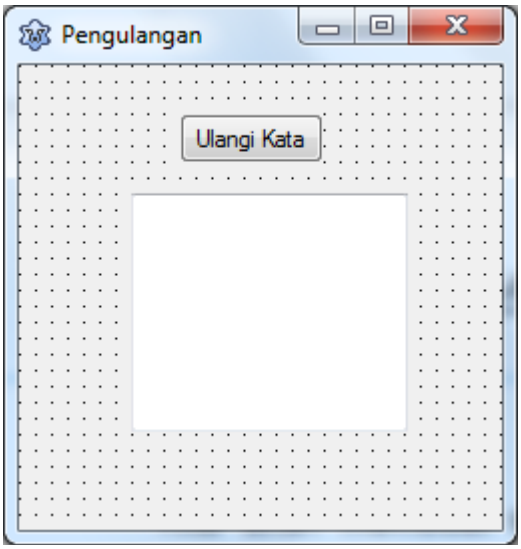
   FOR <variable pengontrol> : = <batas bawah> **downto** <batas atas> **do** <pernyataan>

Contoh

```
a:=0;
For i:=1 to 5 do a:=a+1;                akan menghasilkan nilai a = 5

b:=10;
For x:=10 downto 6 do b:=b+1;           akan menghasilkan nilai b = 5

c:=0
For j:=-5 to 5 do c:=c+j;               akan menghasilkan nilai c = 0
```

Latihan 1

Kita akan membuat sebuah program sederhana untuk lebih memahami struktur FOR..DO. Untuk itu buatlah form seperti berikut ini:



Kemudian atur properties-nya sebagai berikut:

Komponen	Properties	Value
Button1	Caption	Pengulangan
Memo1	Lines	

Kemudian klik dua kali pada button1 dan masukkan kode berikut ini:

```
procedure TForm1.Button1Click(Sender: TObject);
var
 i:integer;
begin
  for i:=1 to 6 do
   memo1.Lines.Add('statistika');
end;
```

Atau Tuliskan Kode berikut ini

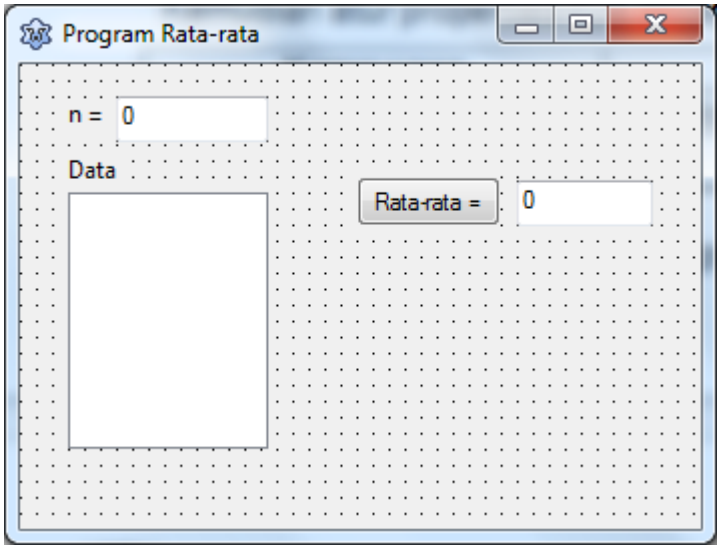
```
procedure TForm1.Button1Click(Sender: TObject);
var
 i:integer;
begin
  for i:=8 downto 3  do
   memo1.Lines.Add('statistika');
end;
```

Setelah dieksekusi akan menghasilkan seperti berikut ini

Hasil dari kedua kode tersebut bernilai sama meskipun nilai awal dan nilai akhir yang diinputkan berbeda, karena didalam program FOR..DO diatas yang dilihat adalah jumlah pengulanggannya.

Latihan 2

Dalam latihan kali ini kita akan membuat program untuk menghitung nilai rata-rata siswa dari mata kuliah tertentu. Input : banyak siswa, nilai-nilainya. Output : Nilai rata-rata. Dalam latihan kali ini kita akan menggunakan Blok statement pada struktur For..Do. Untuk itu buatlah form dengan desain seperti berikut ini:



Kemudian atur properties-nya sebagai berikut:

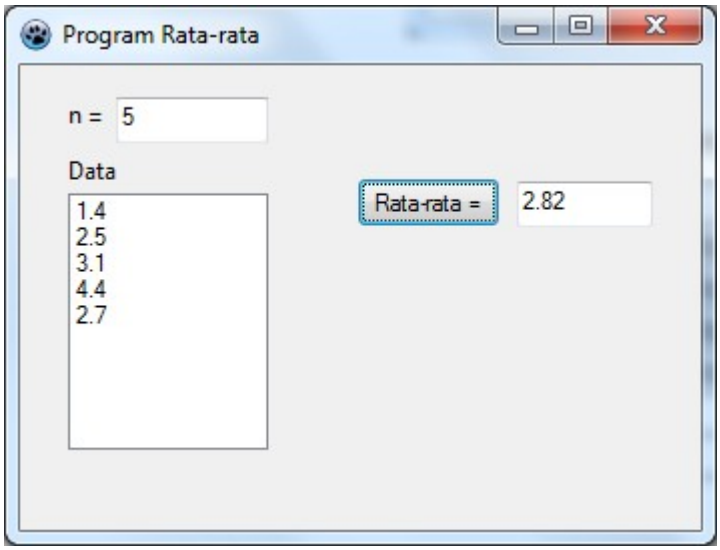
Komponen	Properties	Value
Button1	Caption	Rata-rata =
ListBox1	Caption	
Label1	Caption	n =
Label2	Caption	Data
Edit1	Text	0
Edit2	Text	0

Kemudian klik dua kali pada button1 dan masukkan kode berikut ini:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  n,i:integer;
  rata2:real;
begin
 //input data
 n:=strtoint(edit1.text);
 for i:=1 to n  do
     listbox1.Items.Add(inputbox('Input Data', 'X' + inttostr(i),''));
 //perhitungan
 rata2:=0;
```

```
 for i:=1 to n do
     rata2:=rata2+strtofloat(listbox1.Items[i-1])/n;
 edit2.text:=floattostr(rata2);
end;
```

Setelah dieksekusi akan menghasilkan seperti berikut ini



Bagian 2

PENGULANGAN DENGAN KONDISI

A. Struktur While … Do

Struktur pengulangan WHILE .. DO tidak ditentukan berapa banyak pengulangan dilakukan akan tetapi menggunakan ekspresi logika untuk menghentikan pengulangan. Berbeda dengan struktur FOR .. DO yang menghentikan pengulangan hanya dengan batas pencacahannya saja. Dalam struktur WHILE .. DO pengulangan akan terus dilakukan selama ekspresi logika terpenuhi dan akan berhenti pada saat ekspresi logika tidak terpenuhi. Ekspresi logika yang digunakan dalam WHILE .. DO sama dengan ekspresi logika yang digunakan dalam struktur IF..THEN. Bentuk umum struktur WHILE .. DO adalah sebagai berikut

```
WHILE <ekspresi logika> DO
        Begin
            <pernyataan>;
            …..
        End;
```

Blok statement dalam struktur WHILE .. DO akan dibutuhkan karena jarang sekali pernyataan dalam struktur ini hanya terdiri dari satu pernyataan.

Contoh

```
a:=0;
While a<=5 do
        Begin
            memo1.Lines.Add('statistika');
            a:=a+1;
        End;
```

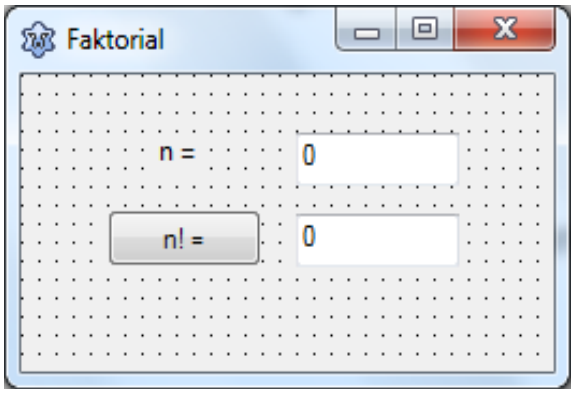
Pernyataan a:=a+1 diperlukan agar pengulangan bisa berhenti.

Latihan 3

Dalam latihan kali ini kita akan membuat program untuk menghitung faktorial sebuah bilangan bulat. Input : bilangan bulat positif. Output : faktorialnya. Rumus faktorial adalah sebagai berikut

$$n! = 1*2*...*(n-2)*(n-1)*n$$

Untuk itu buatlah sebuah Form dengan desain sebagai berikut



Kemudian atur properties-nya sebagai berikut

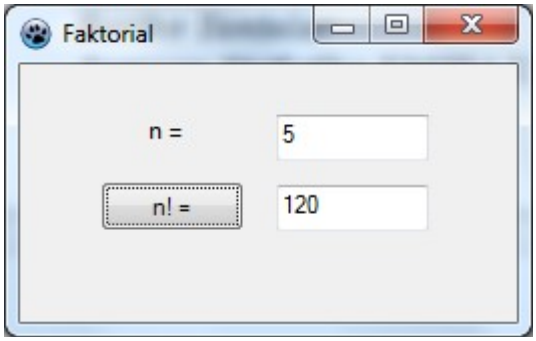
Komponen	Properties	Value
Button1	Caption	n! =
Label1	Caption	n =
Edit1	Text	0
Edit2	Text	0
Form1	Caption	Faktorial

Kemudian klik dua kali pada button dan masukkan kode berikut ini

```
procedure TForm1.Button1Click(Sender: TObject);
var
  n,i : integer;
  fak : longint;
begin
```

```
 //input data
 n:=strtoint(edit1.text);
 //perhitungan
 if n < 0 then
   messagedlg('Angka yang anda masukkan negatif',mterror,[mbYes],0)
 else
  begin
   i:=1;
   fak:=1;
   if n>0 then
     while i<=n do
       begin
         fak:=fak*i;
         i:=i+1;
       end;
  end;
 edit2.text:=floattostr(fak);
end;
```

Setelah program dieksekusi akan menghasilkan jendela sebagai berikut



B. Struktur Repeat .. Until

Sama dengan struktur WHILE .. DO, struktur pengulangan REPEAT .. UNTIL juga tidak ditentukan berapa banyak pengulangan dilakukan akan tetapi menggunakan ekspresi logika untuk menghentikan pengulangan. Dalam struktur REPEAT .. UNTIL pengulangan akan berhenti pada saat ekspresi logika terpenuhi dan akan terus dilakukan selama ekspresi logika tidak terpenuhi. Ekspresi logika yang digunakan dalam REPEAT .. UNTIL sama dengan ekspresi logika yang digunakan dalam struktur IF..THEN. Bentuk umum struktur REPEAT .. UNTIL adalah sebagai berikut

```
REPEAT
        <pernyataan>;
        ….
UNTIL <ekspresi logika>
```

Tidak seperti struktur WHILE .. DO yang memerlukan Blok statement dalam pengulangannya, Struktur REPEAT .. UNTIL tidak membutuhkan Blok Statement meskipun lebih dari satu pernyataan.

Contoh

```
a:=0;
Repeat
    memo1.Lines.Add('statistika');
    a:=a+1;
Until a>5;
```

Pernyataan a:=a+1 diperlukan agar pengulangan bisa berhenti.

Latihan 4

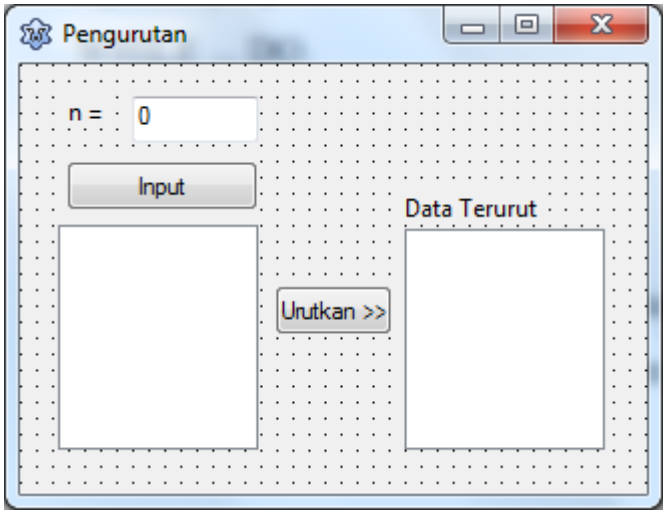
Dalam latihan kali ini kita akan memodifikasi program pada Latihan 3 dengan mengubah kode menjadi sebagai berikut

```
procedure TForm1.Button1Click(Sender: TObject);
var
  n,i : integer;
  fak : longint;
begin
 //input data
 n:=strtoint(edit1.text);
 //perhitungan
 if n < 0 then
    messagedlg('Angka yang anda masukkan negatif',mterror,[mbYes],0)
 else
  begin
    i:=1;
    fak:=1;
    if n>0 then
       repeat
           fak:=fak*i;
           i:=i+1;
        until i>n;
  end;
 edit2.text:=floattostr(fak);
end;
```

Apabila program dieksekusi maka akan menghasilkan nilai yang sama dengan WHILE ... DO.

Latihan 5

Pada latihan kali ini kita akan membuat sebuah program untuk mengurutkan n buah data dengan input banyak data (n) dan nilai-nilai datanya. Untuk itu buatlah sebuah form dengan desain sebagai berikut



Kemudian atur properties-nya sebagai berikut

Komponen	Properties	Value
Button1	Caption	Input
Button2	Caption	Urutkan >>
Label1	Caption	n =
Label2	Caption	Data Terurut
Edit1	Text	0
ListBox1	Caption	
ListBox1	Caption	

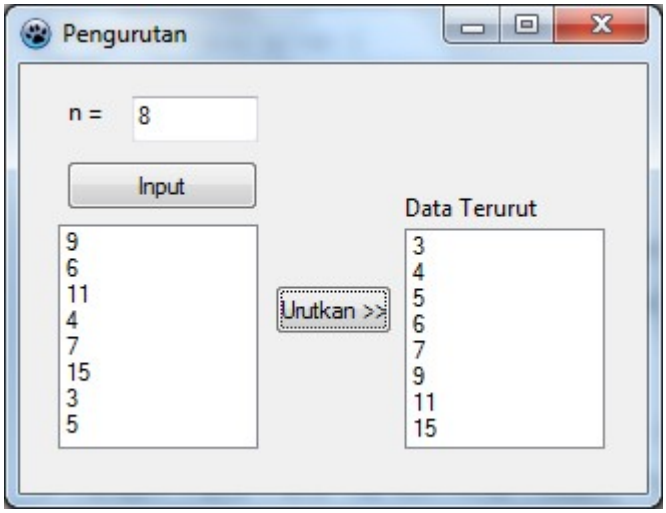
Kemudian klik dua kali pada button1 dan masukkan kode berikut ini

```
procedure TForm1.Button1Click(Sender: TObject);
var i,n:integer;
begin
   n:=strtoint(edit1.text);
   for i:=1 to n do
        listbox1.items.add(inputbox('Input Data', 'X' + inttostr(i),''));
end;
```

Dan klik dua kali pada button2 dan masukkan kode berikut ini

```
procedure TForm1.Button2Click(Sender: TObject);
var i,j,n:integer;
   a,b:real;
   temp:string;
begin
   n:=strtoint(edit1.text);
   listbox2.items:=listbox1.items;
   for i:=1 to n-1 do
   for j:=i+1 to n do
      begin
       a:=strtofloat(listbox2.Items[i-1]);
       b:=strtofloat(listbox2.Items[j-1]);
       if a > b then
         begin
          temp:=listbox2.items[i-1];
          listbox2.items[i-1]:=listbox2.items[j-1];
          listbox2.items[j-1]:=temp;
         end;
      end;
end;
```

Setelah program dieksekusi akan menghasilkan jendela sebagai berikut



C. Soal-soal

1. Buatlah sebuah program untuk menghitung varians dan simpangan baku dari suatu data. Input adalah banyak data (n) dan nilai-nilai datanya. Outputnya adalah varians dan simpangan baku. Rumus varians adalah sebagai berikut

$$S^2 = \frac{\sum_{i=1}^{n} (X_i - \bar{X})^2}{n-1}$$

   dan simpangan baku adalah akar dari varians.

2. Buatlah sebuah program untuk menghitung nilai fungsi distribusi kumulatif Binomial. Dengan input banyak percobaan (n), banyak sukses (k) dan peluang sukses (p). Output adalah nilai peluang kumulatif binomial. Rumus untuk distribusi kumulatif Binomial adalah sebagai berikut:

$$P(X \le k) = \sum_{x=0}^{k} \binom{n}{x} p^x (1-p)^{n-x}$$

MODUL 5

VARIABEL BERINDEKS

PENDAHULUAN

Dalam modul ini akan dipelajari mengenai variabel berindeks yang disebut dengan array, mendefinisikan tipe, menggunakan set dan tipe variabel record.

Modul ini terdiri atas dua bagian yaitu pada bagian pertama mengenai ariabel berindeks array yang terdiri atas array, array multidimensi dan array dinamis.

Pada bagian kedua akan dijelaskan membuat tipe variabel, menggunakan set dan membuat record sebagai tipe variabel.

Semua variabel yang kita gunakan pada modul sebelumnya hanya memuat satu ruang saja, padahal seringkali kita membutuhkan sebuah variabel dengan banyak tempat. Misalnya saja variabel X mempunyai nilai-nilai 1, 3, 7, 2. Biasanya pendefinisian nilai-nilai dari variabel tersebut menggunakan indeks seperti berikut ini: X1 = 1, X2 = 3, X3 = 7 dan X4 = 2. Dapat dikatakan bahwa variabel X mempunyai 4 ruang yang tempatnya didefiniskan sebagai indeks. Dalam Bahasa Pemrograman Lazarus telah disediakan tipe variabel berindeks seperti ini yang disebut dengan array.

Bagian 1

ARRAY

A. Array

Array adalah sebuah tipe variabel yang memiliki banyak ruang. Array juga disebut sebagai variabel berindeks. Setiap indeks yang akan diguanakan dalam variabel bertipe array harus didefiniskan pada bagian deklarasi. Umumnya indeks didefiniskan sebagai sebuah rentang nilai integer. Dalam satu variabel array semua ruang yang ada harus memiliki tipe yang sama. Secara umum cara pendeklarasian variabel array adalah sebagai berikut

Nama_Variabel : Array[nilai_awal..nilai_akhir] of tipe_variabel;

Contoh

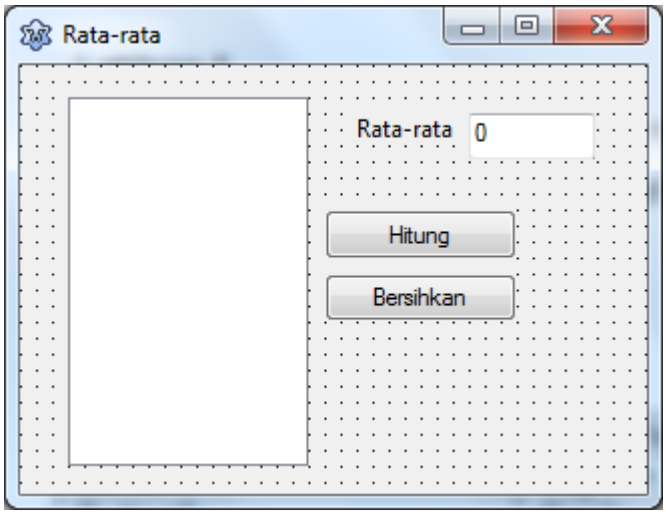
Var	
Nama : Array[1..10] of string	memiliki 10 ruang dengan tipe string
IPK : Array[5..15] of real	memiliki 11 ruang dengan tipe real
nomor : Array[0..5] of integer	memiliki 6 ruang dengan tipe integer

Perlu diketahui bahwa hanya indeks yang didefinisikan yang dapat digunakan pada variabel array tersebut. Misalnya untuk variabel IPK pada contoh diatas memiliki indeks dari 5 hingga 15 tetapi tidak memiliki indeks 1 sampai 4, variabel Nama tidak memiliki indeks 0 dan variabe; nomor tidak memiliki indeks lebih dari 5. Artinya apabila kita mendefinisikan IPK[3] atau Nama[0] atau nomor[7] akan menghasilkan error pada saat dieksekusi.

Beberapa objek dalam Lazarus merupakan variabel bertipe array seperti ListBox, ComboBox dan StringGrid. Semua objek bertipe array yang ada dalam Lazarus memiliki tipe variabel string dengan nilai awal indeks adalah nol (0) dan nilai akhir adalah banyaknya data yang diinputkan.

Latihan 1

Dalam latihan kali ini kita akan membuat program untuk menghitung rata-rata menggunakan array. Untuk itu buatlah sebuah Form dengan desain sebagai berikut



Kemudian atur propertiesnya sebagai berikut:

Komponen	Properties	Value
Button1	Caption	Hitung
Button2	Caption	Bersihkan
Memo1	Lines	
Edit1	Text	0

Klik dua kali pada button1 dan masukkan kode sebagai berikut

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i,n:integer;
  rata,total:real;
  x:array[1..3] of real;
begin
  total:=0;
  n:=memo1.Lines.Count-1;
```

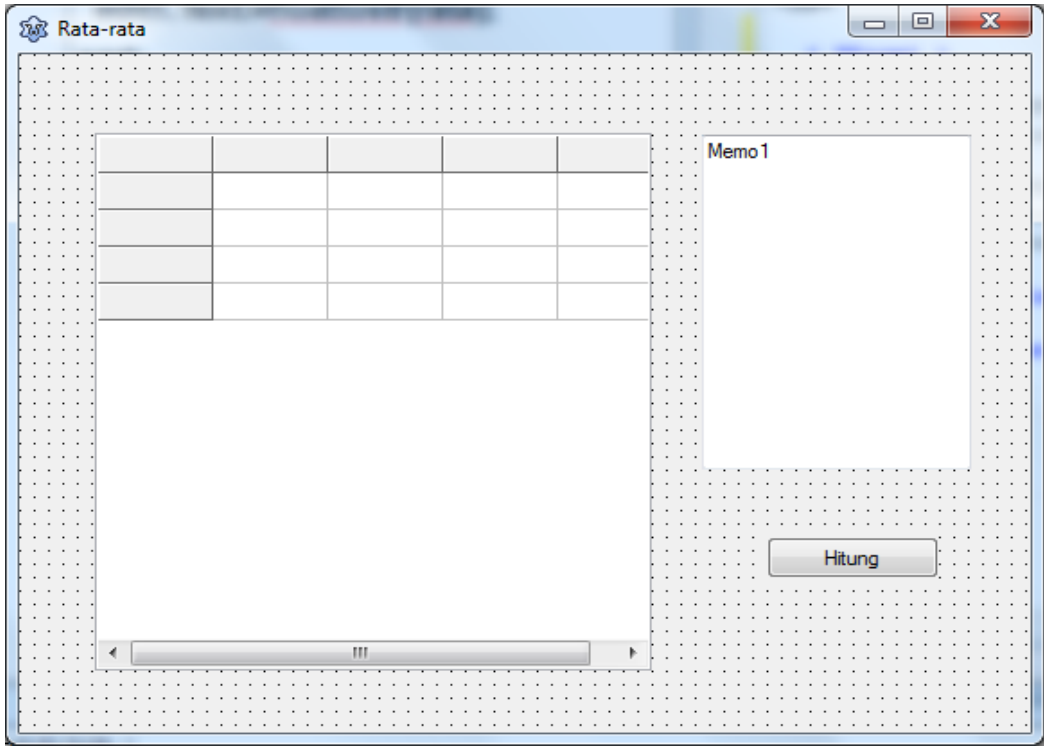
```
 for i:=1 to n do
  begin
    x[i]:=strtofloat(memo1.Lines[i-1]);
  end;
 for i:=1 to n do
  begin
    total:=total+x[i];
  end;
 rata:=total/n;
 edit1.Text:=floattostr(rata);
end;
```

B. Array Multidimensi

Array multidimensi adalah array yang mempunyai dimensi n x m salah satu penggunaan array multidimesi adalah dalam hal operasi matriks.

Latihan 2

Pada latihan kali ini kita akan membuat sebuah program menggunakan stringgrid. Untuk itu buatlah sebuah Form dengan desain seperti gambar berikut



Kemudian atur propertiesnya sebagai berikut:

Komponen	Properties	Value
Button1	Caption	Hitung
Memo1	Lines	
Stringgrid1	Rowcount	5
	Colcount	5
	Goediting	True

Klik dua kali pada button1 dan masukkan kode sebagai berikut

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i,n,j:integer;
  x:array[1..4,1..4] of real;
begin
for i:=1 to 4 do
  begin
```

```
  for j:=1 to 4  do
    begin
     x[i,j]:=strtofloat(stringgrid1.Cells[j,i]);
    end;
  end;
for i:=1 to 4 do
 begin
 for j:=1 to 4 do
 begin
  memo1.Lines.Add('nilai'+'X('+inttostr(i)+','+inttostr(j)+')'+'  adalah
'+floattostr(x[i,j]));
 end;
 end;
end;
```

C. Array Dinamis

Array yang tidak memiliki ukuran atau dimensi yang tetap. Array seperi ini dideklarasikan dengan bentuk seperti berikut:

Array of *tipeDasar*

Latihan 3

Pada Program yang telah dibuat pada Latihan 2 ubahlah kode dalam prosedur **button1** dengan kode sebagi berikut:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i,n,j:integer;
  x:array of array of real;
```

```
begin
setlength(x,10,10);
for i:=1 to 4 do
  begin
   for j:=1 to 4  do
     begin
      x[i,j]:=strtofloat(stringgrid1.Cells[j,i]);
     end;
   end;
for i:=1 to 4 do
 begin
 for j:=1 to 4 do
 begin
   memo1.Lines.Add('nilai'+'X('+inttostr(i)+','+inttostr(j)+')'+'  adalah
'+floattostr(x[i,j]));
 end;
 end;
end;
```

Latihan 4

Dalam latihan kali ini kita akan membuat program untuk mencari inver dari suatu matriks. Syarat sebuah matriks agar diperoleh invernya adalah matriks tersebut harus berbentuk bujursangkar dan non singular (determinannya tidak nol). Dalam program ini akan kita batasi untuk dimensi matriks 5 x 5. Dalam menghitung matriks berdimensi lebih dari 2x2 kita dapat menggunakan algoritma SWEEP yang langkah-langkahnya adalah sebagai berikut:

1. Tentukan $D = a_{kk}$ (untuk setiap $k = 1,2, ..., n$)
2. Kalikan baris ke k matriks A dengan $1/D$

3. Untuk setiap $i \neq k$

   a)Tentukan $B = a_{ik}$

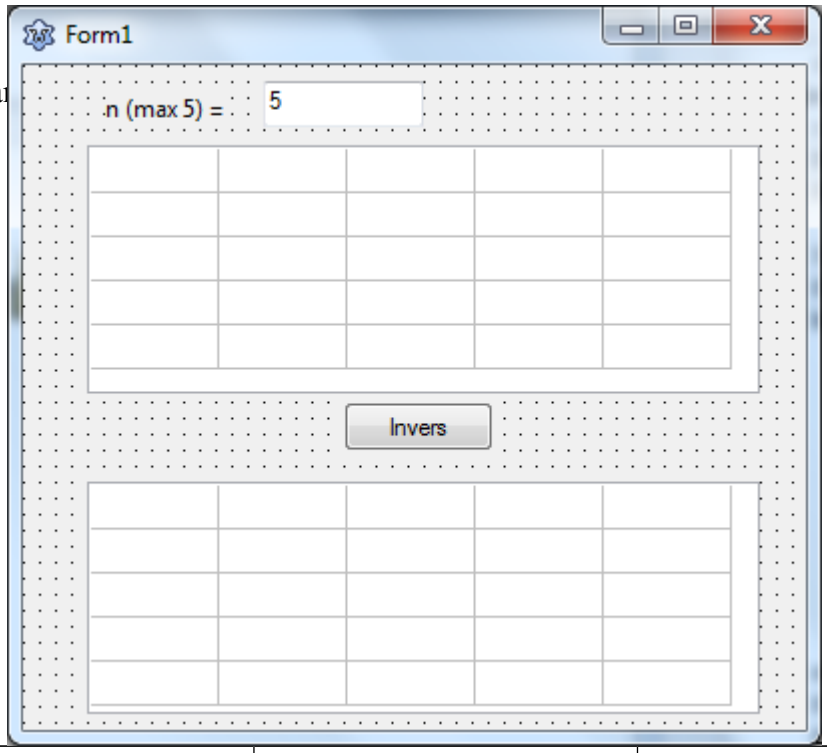
   b)Kurangkan baris ke i dengan B kali baris ke k

   c)Hitung $a_{ik} = -B/D$

4. Hitung $a_{kk} = 1/D$

Buatlah sebuah Form dengan desain sebagai berikut

Kemudian



Komponen	Properties	Value
Button1	Caption	Invers
Label1	Caption	N (max 5)
Edit1	Text	0
Stringgrid1	Rowcount	5
	Colcount	5
	Goediting	True
Stringgrid2	Rowcount	5
	Colcount	5
	Goediting	False

Agar stringgrid secara otomatis mengikuti besar dimensi maka klik dua kali pada Edit1 dan masukkan kode berikut ini

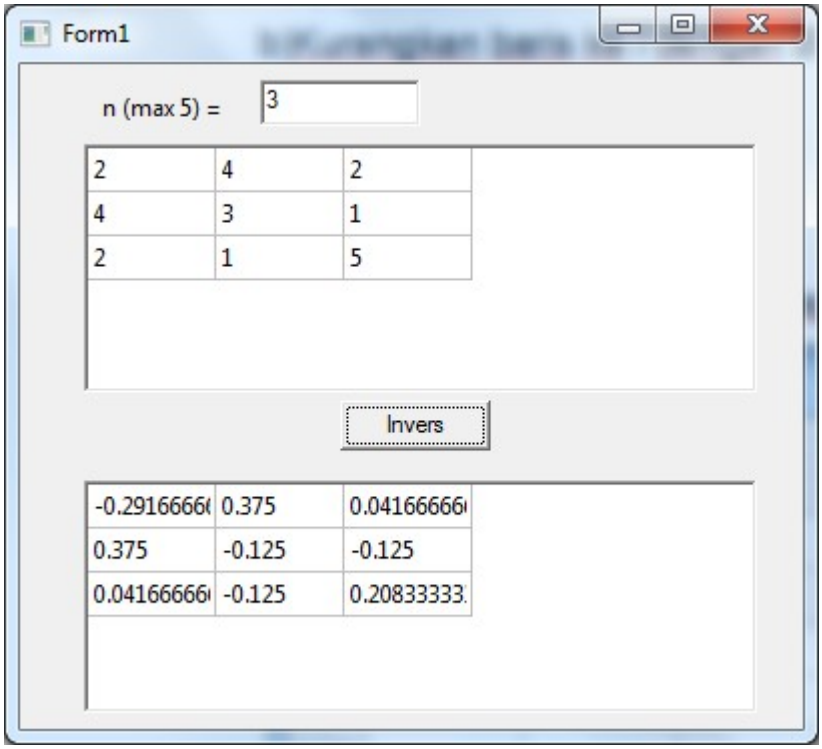
```
procedure TForm1.Edit1Change(Sender: TObject);
var n: integer;
begin
   if (edit1.text<>'') then
   begin
     n:=strtoint(edit1.text);
     if n<6 then
     begin
       stringgrid1.ColCount:=n;
       stringgrid2.ColCount:=n;
       stringgrid1.rowCount:=n;
       stringgrid2.rowCount:=n;
     end;
   end;
end;
```

Klik dua kali pada button1 dan masukkan kode berikut ini

```
procedure TForm1.Button1Click(Sender: TObject);
var A,B : array[1..5,1..5] of real;
   i,j,k,n : integer;
   d,c:real;
begin
   n:=strtoint(edit1.text);
   for i:=1 to n do
   for j:=1 to n do
      A[i,j]:=strtofloat(stringgrid1.Cells[j-1,i-1]);
   B:=A;
```

```
  for k:=1 to n do
  begin
    d:=B[k,k];
    for j:=1 to n do B[k,j]:=B[k,j]/d;
    for i:=1 to n do
      if i<> k then
      begin
        c:=B[i,k];
        for j:=1 to n do B[i,j]:=B[i,j]-c*B[k,j];
        B[i,k]:=-c/d;
      end;
    B[k,k]:=1/d;
  end;
  for i:=1 to n do
  for j:=1 to n do
    stringgrid2.Cells[j-1,i-1]:=floattostr(B[i,j]);
end;
```

Apabila program dieksekusi maka akan diperoleh hasil sebagai berikut



D. Soal-soal

1. Buatlah program untuk menghitung median menggunakan array.
2. Buatlah sebuah program untuk menghitung rata-rata harmonik dan rata-rata ukur dengan rumus sebagai berikut

$$\text{Rata-rata Harmonik} \quad H = \frac{n}{\sum_{i=1}^{n} 1/X_i}$$

$$\text{Rata-rata Ukur} \quad U = \sqrt[n]{\prod_{i=1}^{n} X_i}$$

3. Buat program untuk mengalikan matriks A = 2 x 3 dan matriks B = 3 x 4
4. Buat program untuk menghitung invers matriks (A) menggunakan Algoritma Adjust dengan matriks gandengan (I). Langkah-langkahnya adalah sebagai berikut
   1. Tentukan $B = a_{kk}$ (untuk setiap $k = 1,2, ..., n$)
   2. Kalikan baris ke k matriks A dengan $1/B$
   3. Untuk setiap $i \neq k$

a)Tentukan $B = a_{ik}$

b)Kurangkan baris ke i dengan B kali baris ke k

4. Hasilnya inversnya adalah matriks gandengannya

Petunjuk: Buat Matriks berukuran n x 2n yang dipartisi menjadi n x n untuk matriks yang akan diinverskan dan n x n untuk matriks gandengan berbentuk matriks identitas.

Bagian 2

TIPE, SET DAN RECORD

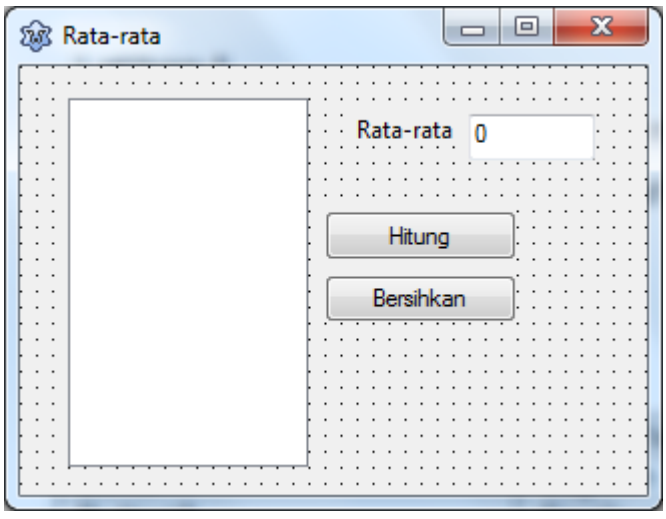
Selain menggunakan tipe data yang sudah ada didalam Lazarus, kita juga dapat mendefinisikan tipe data kita sendiri. Tentunya untuk melakukan hal tersebut kita harus membuat deklarasi tipe variabel yang akan digunakan sebelum mendeklarasikan variabel.

A. Tipe

Kita bisa mendefinisikan tipe data. Tipe yang kita definisikan dapat dipakai untuk pendeklarasian konstanta atau variabel.

Latihan 1

Dalam latihan kali ini kita akan membuat program untuk menghitung rata-rata menggunakan tipe. Untuk itu buatlah sebuah Form dengan desain sebagai berikut



Kemudian atur propertiesnya sebagai berikut:

Komponen	Properties	Value
Button1	Caption	Hitung
Button2	Caption	Bersihkan
Memo1	Lines	
Edit1	Text	0

Klik dua kali pada button1 dan masukkan kode sebagai berikut

```
procedure TForm1.Button1Click(Sender: TObject);
Type
  vektor=array[1..3] of real;
var
  i,n:integer;
  rata,total:real;
  x:vektor;
begin
  total:=0;
  n:=memo1.Lines.Count-1;
  for i:=1 to n do
   begin
     x[i]:=strtofloat(memo1.Lines[i-1]);
   end;
  for i:=1 to n do
   begin
     total:=total+x[i];
   end;
  rata:=total/n;
  edit1.Text:=floattostr(rata);
end;
```

B. Set

Set adalah tipe untuk sejumlah anggota dari suatu tipe sederhana. Misalnya char dan byte. Cara penulisan set adalah sebagai berikut

type

Nama_tipe = set of data_sederhana

Selain itu pada set juga dapat didefinisikan sekelompok bilangan atau sekumpulan karakter. Contoh pendefinisian set adalah sebagai berikut

```
Kata = set of char
Bilangan = set of byte
Angka = set of 0..9
karakter = set of  'a' .. 'z'
```

C. Record

Record adalah tipe data terstruktur yang berisi sejumlah data, dan masing-maisng dapat berbeda tipe. Record banyak digunakan untuk menangani database. Pendeklarasiannya sebagai berikut:

```
Type
  namaTipeRecord = record
      daftar_1: tipe_1;
      daftar_2: tipe_2;
  end;
```

Apabila array dapat menampung banyak variabel dengan tipe sama, records dapat menampung banyak variabel dengan tipe yang berbeda-beda. Variabel-variabel ini disebut dengan 'Fields'.

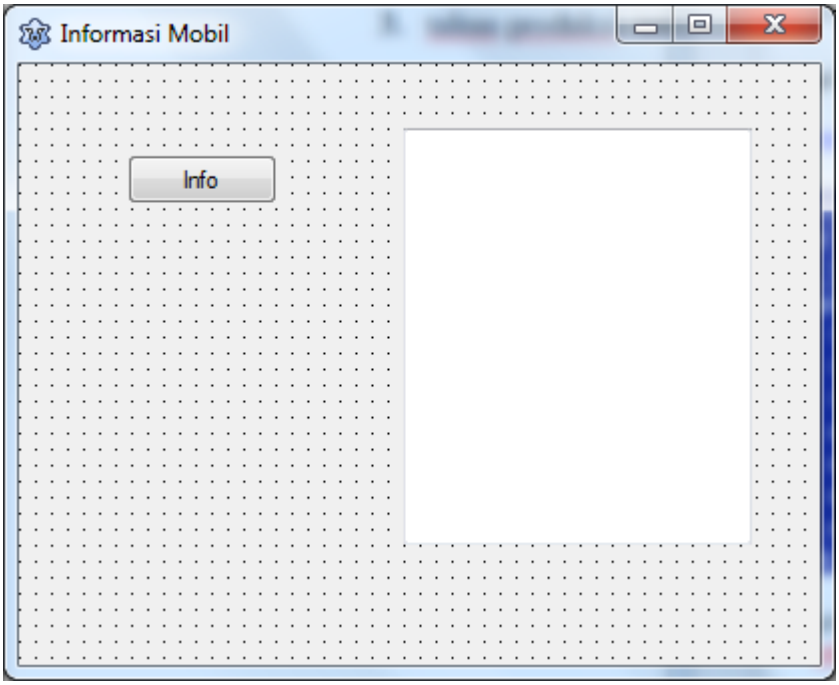
Grup variabel/fields dapat diperlakukan sebagai sebuah unit atau variabel. Kita dapat menggunakan records untuk menempatkan informasi yang berasal dari objek yang sama, misalnya informasi mengenai mobil sebagai berikut:

1. Jenis Mobil: string
2. Ukuran mesin: real
3. tahun produksi: integer

Kita dapat mengumpulkan tipe-tipe variabel berbeda ini ke dalam satu records yang merepresentasikan sebuah mobil.

Latihan 2

Kta akan membuat program untuk menampilkan informasi dari mobil seperti pada contoh sebelumnya. Buatlah sebuah Form dengan desain seperti berikut ini



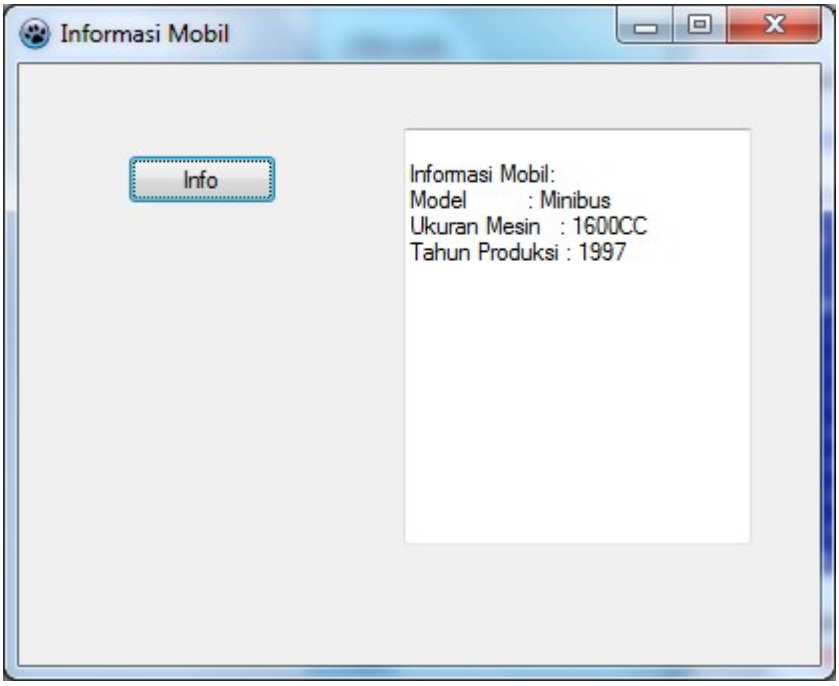
Pada bagian **type** dalam source editor tambahkan baris kode berikut:

```
TCar = record
    ModelName: string;
    Engine: Single;
    ModelYear: Integer;
end;
```

Kemudian klik dua kali button1 dan tulis kode berikut

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Car: TCar;
begin
  Car.ModelName:=Inputbox('Info Mobil','Input Model Mobil: ','');
  Car.Engine:=strtofloat(Inputbox('Info Mobil','Input Ukuran Mesin: ',''));
   Car.ModelYear:=strtoint(Inputbox('Info Mobil','Input Tahun Produksi:
','','));
  memo1.Lines.add('Informasi Mobil: ');
  memo1.Lines.add('Model          : ' + Car.ModelName);
  memo1.Lines.add('Ukuran Mesin   : ' + floattostr(Car.Engine) + 'CC');
  memo1.Lines.add('Tahun Produksi : ' + inttostr(Car.ModelYear));
end;
```

Apabila program di eksekusi akan menghasilkan seperti gambar dibawah ini



Dalam contoh ini kita dapat mendefinisikan sebuah tipe baru (records) menggunakan kata 'type'. Perhatikan kode program berikut ini:

```
type TCar = record
    ModelName: string;

    Engine: Single;

    ModelYear: Integer; end;
```

Dalam program tersebut kita menambahkan huruf T pada kata Car untuk mengindikasikan bahwa pada ini adalah sebuah tipe dan bukan sebuah variabel. Nama variabel seperti: Car, Hour, UserName, tetapi nama type seperti: TCar, THouse, and TUserName. Hal ini merupakan bahasa standar dalam Pascal.

Waktu kita membutuhkan sebuah tipe baru, kemudian kita harus mendeklarasikan sebuah variabel menggunakan tipe tersebut kita dapat menuliskannya sebagai berikut:

```
var
   Car: TCar;
```

Apabila kita akan menyimpan sebuah nilai kedalam salah satu variabel/fields tadi, maka kita dapat menuliskannya sebagai berikut

```
Car.ModelName
```

D. Soal-soal

1. Buatlah sebuah program untuk menginputkan data mengenai mahasiswa sebagai berikut

Nama	: string
NPM	: integer
Alamat	: String
IPK	: real
Jumlah SKS	: integer

Petunjuk: Gunakan records

2. Buatlah sebuah program untuk menginputkan data mengenai pegawai sebagai berikut

Nama	: string
Alamat	: String
Gaji	: real
Absen	: integer

Petunjuk: Gunakan Tipe dan records

MODUL 6

PROSEDUR DAN FUNGSI

PENDAHULUAN

Dalam modul ini akan dipelajari mengenai prosedur dan fungsi yang meliputi bagaimana cara membuat dan cara menjalankan prosedur dan fungsi. Selain itu juga dipelajari bagaimana memanfaatkan prosedur dan fungsi yang telah ada.

Modul ini terdiri atas dua bagian yaitu pada bagian pertama mengenai bagaimana membuat prosedur dan menggunakannya di dalam program. Sedangkan pada bagian kedua mengenai bagaimana membuat fungsi dan menggunakannya pada program.

Prosedur dan fungsi merupakan bagian dari suatu program atau dengan kata lain disebut sebagai sub program. Prosedur dan fungsi dalam penulisannya dilambangkan dengan satu kata. Prosedur merupakan potongan program yang berdiri sendiri yang tidak membutuhkan operasi dari objek lain. Sebaliknya fungsi merupakan potongan program yang bergantung pada operasi dari objek lain. Oleh karena itu fungsi selalu mengembalikan sebuah nilai dalam prosesnya sedangkan prosedur tidak mengambalikan nilai dalam prosesnya.

Bagian 1

PROSEDUR

A. Procedure

Procedure adalah bagian program yang berisi serangkaian langkah-langkah tertentu. Prosedur merupakan potongan program yang berdiri sendiri yang tidak membutuhkan operasi dari objek lain. Kita tidak dapat menuliskan operasi matematika terhadap prosedur. Prosedur dideklarasikan di awal program sebelum program utama (main program) dilaksanakan. Prosedur juga tidak akan dijalankan apabila tidak dipanggil didalam main program. Secara umum sebuah prosedur dapat berjalan pada program utama, di dalam fungsi maupun di dalam prosedur lainnya. Deklarasi procedure yang paling sederhana adalah:

```
Procedure <nama procedure>;
Begin
        …perintah/pernyataan;
end;
```

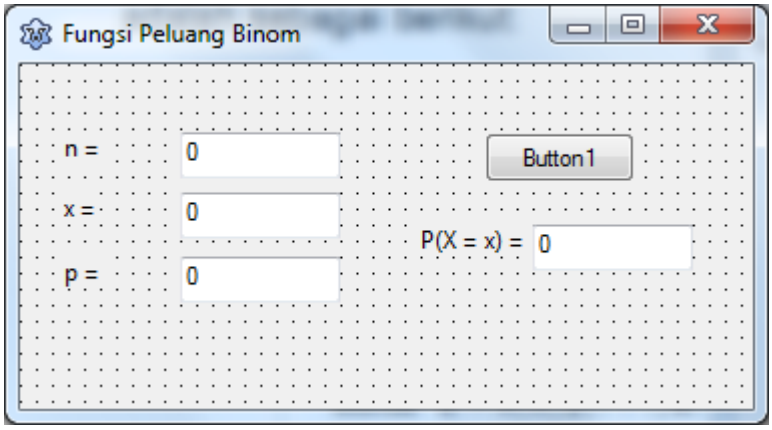
Latihan 1

Dalam latihan kali ini kita akan membuat program untuk menghitung distribusi peluang binom dengan menggunakan procedure. Rumus fungsi peluang binomial adalah sebagai berikut:

$$P(X = k) = p(k) = \binom{n}{k} p^k (1 - p)^{n-k}$$

untuk $k = 0, 1, 2, \ldots, k.$ dan $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

Buatlah sebuah Form dengan desain seperti berikut ini



Kemudian aturlah properties sebagai berikut

Komponen	Properties	Value
Button1	Caption	Hitung
Label1	Caption	N =
Label2	Caption	X =
Label3	Caption	P(X=x) =
Edit1	Text	0
Edit2	Text	0
Edit3	Text	0

Pada Source editor tuliskan kode berikut ini setelah implementation

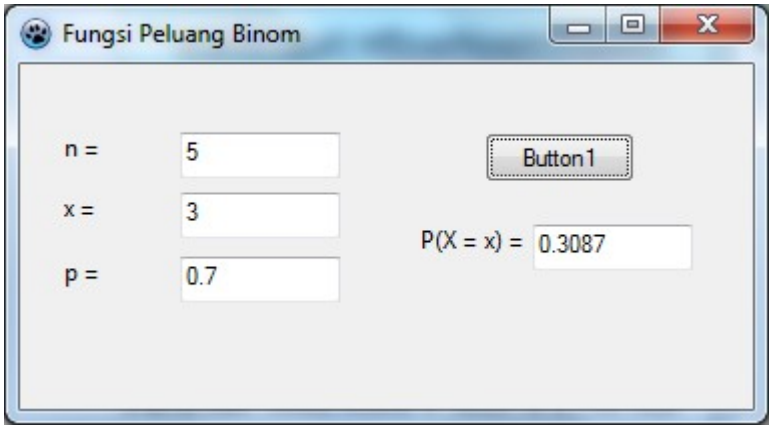
```
procedure faktorial(x:integer; var fak:longint);
var i:integer;
begin
  fak:=1;
  if x>0 then
```

```
  begin
      for i:=1 to x do fak:=fak*i;
  end;
end;

procedure kombinasi(n,x:integer; var kom:real);
var fn,fx,fnx:longint;
begin
    faktorial(n,fn);
    faktorial(x,fx);
    faktorial(n-x,fnx);
    kom:=fn/(fnx*fx);
end;
```

Klik dua kali pada button dan masukkan kode sebagai berikut

```
procedure TForm1.Button1Click(Sender: TObject);
var n,x:integer;
   hasil,p,knx:real;
begin
   n:=strtoint(edit1.text);
   x:=strtoint(edit2.text);
   p:=strtofloat(edit3.text);
   kombinasi(n,x,knx);
   hasil:=knx*exp(x*ln(p))*exp((n-x)*ln(1-p));
   edit4.text:=floattostr(hasil);
end;
```

Setelah dieksekusi maka akan keluar tampilan seperti berikut ini



B. Soal-soal

Langganan PDAM dikota A dielompokkan menjadi 3 golongan. Setiap golongan mempunyai tarif per meter kubik dan harga sewa meteran per bulan.

Golongan	Tarif/m^3	sewa meteran
A	Rp 300	Rp 5000
B	Rp 250	Rp 3500
C	Rp 175	Rp 2500

Buat program yang akan menghitung biaya total langganan, dengan input : nama, alamat, pemakaian dan golongan.

langkah-langkah :

Buatlah procedure berikut :

1. Baca data
2. Hitung
3. Judul dipanggil oleh procedure cetak
4. Cetak

Pada procedure hitung, buatlah susunan berikut:

1. menghitung tarif per m^3 dan sewa meteran (pakai struktur case)
2. menghitung jumlah pembayaran
3. menghitung total pembayaran

Bagian 2
FUNGSI

A. Fungsi

Fungsi adalah bagian program yang berisi serangkaian langkah-langkah tertentu yang menghasilkan nilai. Fungsi merupakan potongan program yang tidak berdiri sendiri yang membutuhkan operasi dari objek lain. Kita dapat menuliskan operasi matematika terhadap fungsi karena fungsi mengembalikan sebuah nilai. Seperti halnya prosedur, fungsi juga dideklarasikan di awal program sebelum program utama (main program) dilaksanakan dan tidak akan dijalankan apabila tidak dipanggil didalam main program. Secara umum sebuah fungsi dapat berjalan pada program utama, di dalam fungsi maupun di dalam prosedur lainnya. Pendeklarasian Fungsi adalah sebagai berikut:

```
Function <nama fungsi>;
        Begin
                …
                …perintah/pernyataan
                …
        end;
```

Contoh

```
Function Tambah(x,y:integer):integer;
 Begin
  Tambah:=x+y;
 End;
```

Pada sebuah fungsi jika parameter nilai hanya dilewatkan pada parameter aktual(parameter sewaktu fungsi atau prosedur dipanggil) maka perubahannya hanya terjadi didalam fungsi sedangkan Nilai yang memanggil tidak akan berpengaruh. Ini disebut dengan *memanggil fungsi dengan parameter nilai*.

Pada sebuah fungsi jika parameter varaibel dilewatkan pada parameter aktual(parameter sewaktu fungsi atau prosedur dipanggil) maka perubahannya tidak hanya terjadi didalam fungsi juga Nilai yang memanggil . Ini disebut dengan *memanggil fungsi dengan parameter varaibel*.

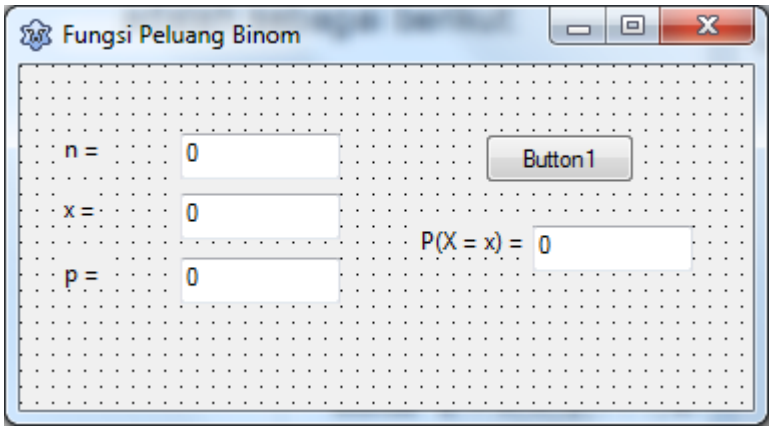
Latihan 1

Dalam latihan kali ini kita akan membuat program untuk menghitung distribusi peluang kumulatif distribusi binomial dengan menggunakan fungsi. Rumus fungsi peluang kumulatif binomial adalah sebagai berikut:

$$P(X \le x) = \sum_{k=0}^{x} \binom{n}{k} p^k (1-p)^{n-k}$$

untuk $k = 0, 1, 2, \ldots, k.$ dan $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

Buatlah sebuah Form dengan desain seperti berikut ini



Kemudian aturlah properties sebagai berikut

Komponen	Properties	Value
Button1	Caption	Hitung
Label1	Caption	N =
Label2	Caption	X =
Label3	Caption	P(X<=x) =
Edit1	Text	0
Edit2	Text	0
Edit3	Text	0

Pada Source editor tuliskan kode berikut ini setelah implementation

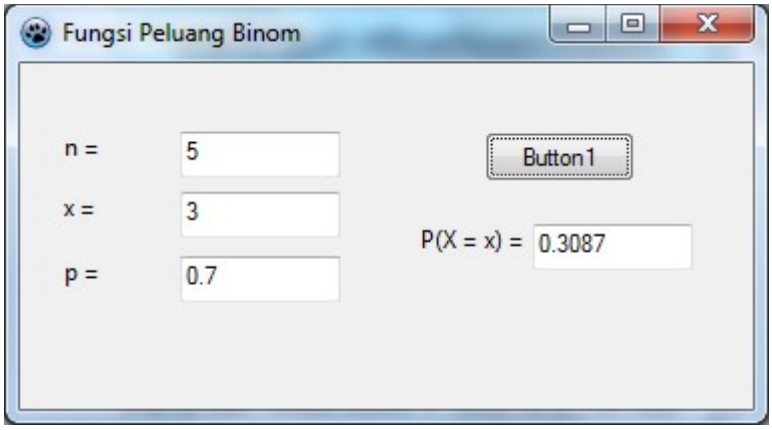
```
function faktorial(x:integer):longint;
var i:integer;
begin
  faktorial:=1;
  if x>0 then
  begin
     for i:=1 to x do faktorial:=faktorial*i;
  end;
end;

function kombinasi(n,x:integer):real;
begin
   kombinasi:=faktorial(n)/(faktorial(n-x)*faktorial(x));
end;
```

Klik dua kali pada button dan masukkan kode sebagai berikut

```
procedure TForm1.Button1Click(Sender: TObject);
```

```
var n,x,k:integer;
    hasil,p:real;
begin
   n:=strtoint(edit1.text);
   x:=strtoint(edit2.text);
   p:=strtofloat(edit3.text);
   hasil:=0;
   for k:=1 to x do
        hasil:=hasil+kombinasi(n,k)*exp(k*ln(p))*exp((n-k)*ln(1-p));
   edit4.text:=floattostr(hasil);
end;
```

Setelah dieksekusi maka akan keluar tampilan seperti berikut ini



B. Fungsi-fungsi Statistika

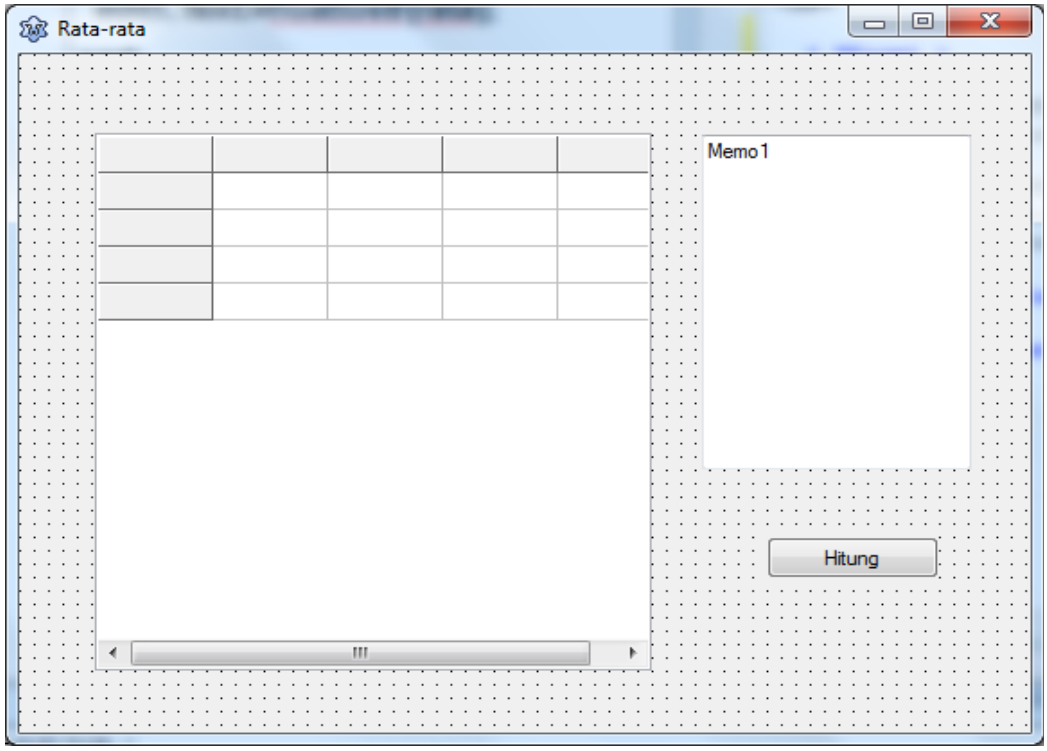
Lazarus telah menyediakan fungsi-fungsi khusus statistika yang bisa digunakan untuk menghitung rata-rata, varians, kuadrat jumlah dan lain-lain. Caranya dengan menuliskan unit **math** pada header program sebagai berikut:

```
Unit1;
```

```
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms, Dialogs, StdCtrls, Grids, math;

type
 TForm1 = class(TForm)
  StringGrid1: TStringGrid;
  Button1: TButton;
  Memo1: TMemo;
  procedure Button1Click(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;
```

Latihan 2

Pada latihan kali ini kita akan membuat sebuah program menggunakan Unit **math**. Untuk itu buatlah sebuah Form dengan desain seperti gambar berikut



Kemudian atur propertiesnya sebagai berikut:

Komponen	Properties	Value
Button1	Caption	Hitung
Memo1	Lines	
Stringgrid1	Rowcount	5
	Colcount	5
	Goediting	True

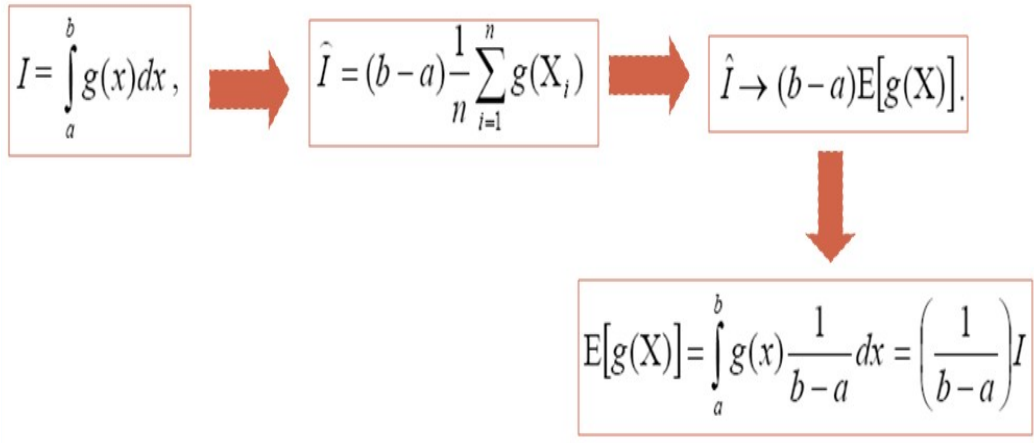
Klik dua kali pada button1 dan masukkan kode sebagai berikut

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i,n:integer;
  total,rata,varians:real;
  x:array of double;
begin
setlength(x,10);
```

```
n:=stringgrid1.RowCount-1;
for i:=1 to n do
begin
  x[i]:=strtofloat(stringgrid1.Cells[1,i]);
end;
rata:=mean(x);
memo1.Lines.Add('rata-rata nya adalah '+floattostr(rata));
end;
```

C. Fungsi Integral

Dalam ilmu statistika banyak dibutuhkan perhitungan menggunakan integral terutama dalam menghitung nilai distribusi peluang kontinu. Didalam Lazarus fungsi integral tidak disediakan sehingga kita harus menggunakan pendekatan numerik dengan metode integrasi monte carlo. Algoritma integrasi monte carlo dapat dilihat pada bagan berikut ini



Gambar 1. Bagan Metode Integrasi Monte Carlo

Keterangan:

a : batas bawah integral

b : batas atas integral

n : banyak pengulangan

g(X) : fungsi yang akan diintegralkan

Dilihat dari alur bagan tersebut terlihat bahwa semakin besar nilai n maka pendekatan ini akan semakin baik. Dengan demikian kita harus menentukan besaran dari *n* agar pengulangannya tidak terlalu lama dan hasilnya cukup baik. Input dari metode integrasi monte carlo tersebut adalah batas bawah, batas atas, banyaknya pengulangan.

Latihan 3

Dalam latihan kali ini kita akan membuat sebuah program untuk menghitung nilai distribusi kumulatif dari Distribusi Normal Baku dengan rumus sebagai berikut

$$P(Z \le z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx$$

Agar persamaan diatas dapat dihitung kita akan menghilangkan unsur '-~' dengan membagi menjadi dua bagian seperti persamaan berikut ini

1. Jika nilai z >= 0 maka

$$P(Z \le z) = \int_{-\infty}^{0} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx + \int_{0}^{z} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx$$

$$P(Z \le z) = 0.5 + \int_{0}^{z} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx$$

2. Jika nilai z < 0 maka

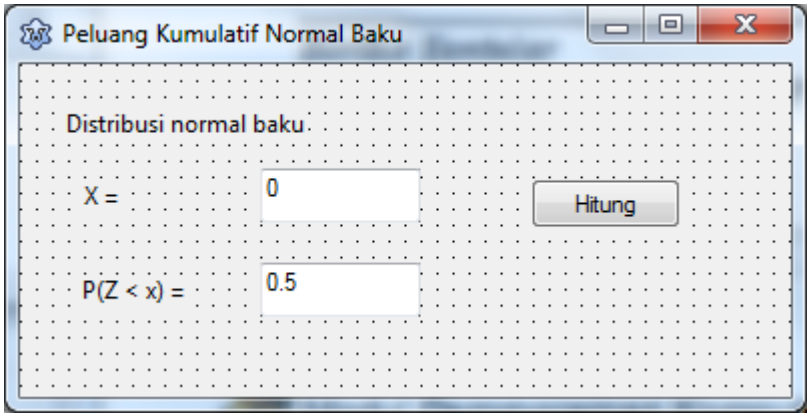
$$P(Z > z) = \int_{z}^{0} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx + \int_{0}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx$$

$$P(Z > z) = \int_{z}^{0} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx + 0.5$$

$$P(Z \le z) = 1 - P(Z > z)$$

$$P(Z \le z) = 0.5 - \int_{z}^{0} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx$$

Buatlah sebuah form dengan desain sebagai berikut



Kemudian atur propertiesnya sebagai berikut:

Komponen	Properties	Value
Button1	Caption	Hitung
Label1	Caption	X =
Label2	Caption	P(Z < z)
Edit1	Text	0
Edit2	Text	0.5

Tambahkan kode berikut di editor setelah implementation

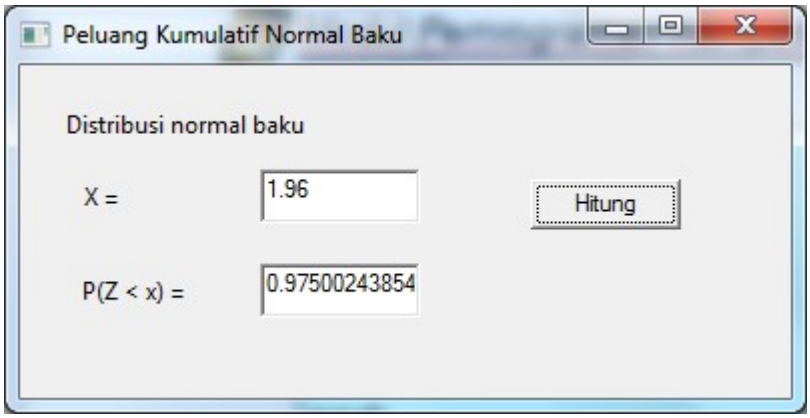
```
procedure distnormal(x:real;var p:real);
var i,n:integer;
   hasil:real;
begin
   n:=1000000;
   hasil:=0;
   for i:= 1 to n do hasil:=hasil+exp(-0.5*sqr((i-1)*x/n));
   if x <=0 then
      p:= 0.5 - exp(-0.5*ln(2*pi))*abs(x)/n*hasil
```

```
  else
     p:= 0.5 + exp(-0.5*ln(2*pi))*x/n*hasil;
end;
```

Klik dua kali pada button1 dan tambahkan kode berikut di editor

```
procedure TForm1.Button1Click(Sender: TObject);
var x,p:real;
begin
  x:=strtofloat(edit1.text);
  distnormal(x,p);
  edit2.text:=floattostr(p);
end;
```

Maka apabila program dieksekusi akan mengahasilkan sebagai berikut



D. Soal-soal

1. Buat program untuk menghitung varians dan kurtosis dengan menggunakan fungsi statistik pada Lazarus?

2. Buatlah sebuah program untuk menghitung fungsi peluang kumulatif dari distribusi hipergeometrik sebagai berikut

$$P(X \le x) = \sum_{k=0}^{x} \frac{\binom{M}{x}\binom{N-M}{n-x}}{\binom{N}{n}}$$

MODUL 7

MEMBUAT DIAGRAM

PENDAHULUAN

Dalam modul ini akan dipelajari mengenai bagaimana membuat diagram di dalam Lazarus. Sebagai bahasa pemrograman visual tentunya Lazarus juga menyertakan sejumlah fungsi untuk membuat grafik.

Modul ini terdiri dari dua bagian yaitu pada bagian pertama mengenai membuat diagram batang menggunakan fungsi barchart. Sedangkan pada bagian kedua adalah menggunakan fungsi Chart untuk membuat berbagai jenis diagram. Dalam statistik deskriptif diagram yang digunakan biasanya diagram batang, diagram lingkaran dan diagram garis. Ketiga jenis diagram ini telah disediakan didalam fungsi Chart.

Bagian 1

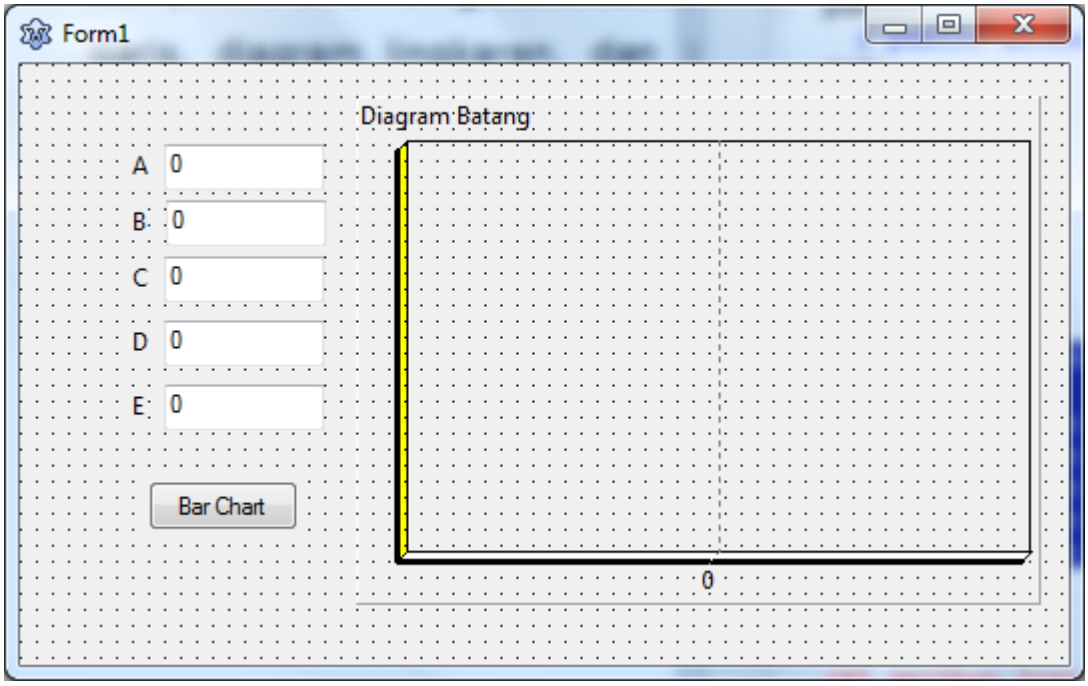
DIAGRAM BATANG

A. BarChart

Komponen BarChart berguna untuk membuat diagram batang dalam Lazarus. Dengan menggunakan komponen ini, penyajian grafik berdasarkan sekumpulan data dapat dilakukan dengan mudah. Komponen BarChart terdapat pada halaman Misc pada Component Palette.

Latihan 1

Dalam latihan kali ini kita akan membuat diagram batang menggunakan fungsi Barchart. Untuk itu buatlah form dengan tampilan sebagai berikut:



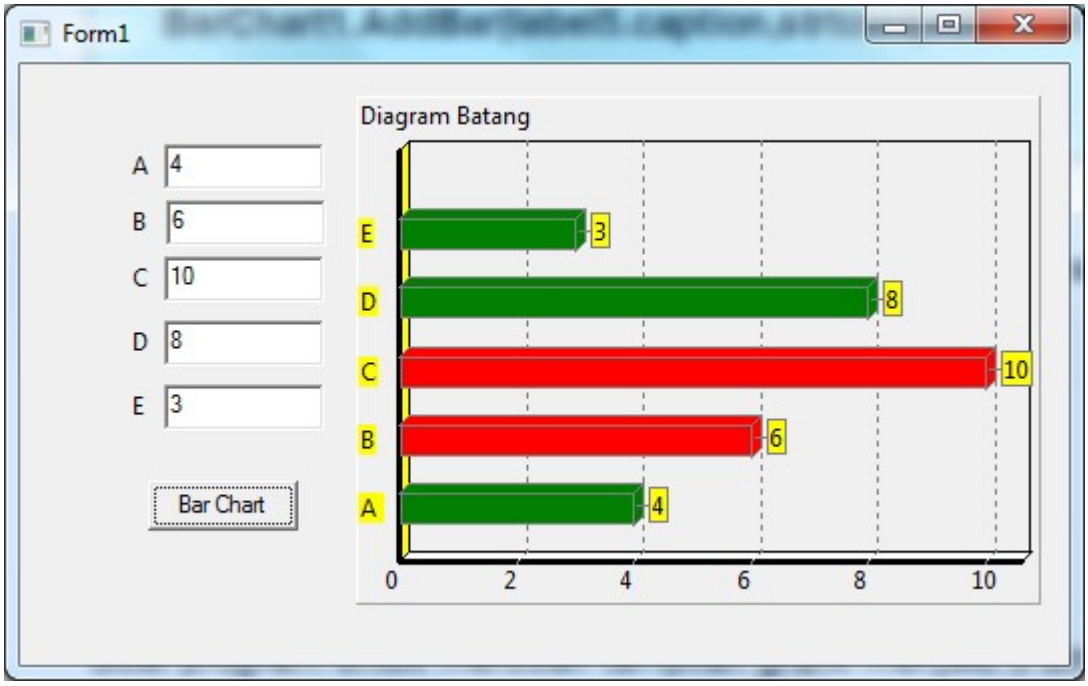
Kemudian atur propertiesnya sebagai berikut:

Komponen	Properties	Value
Button1	Caption	Bar Chart
Edit1	Text	0
Edit2	Text	0
Edit3	Text	0
Edit4	Text	0
Edit5	Text	0
BarChart1	Caption	Diagram Batang

Klik dua kali pada button1 dan masukkan kode sebagai berikut

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    BarChart1.AddBar(label1.caption,strtoint(edit1.text),clGreen);
    BarChart1.AddBar(label2.caption,strtoint(edit2.text),clred);
    BarChart1.AddBar(label3.caption,strtoint(edit3.text),clred);
    BarChart1.AddBar(label4.caption,strtoint(edit4.text),clGreen);
    BarChart1.AddBar(label5.caption,strtoint(edit5.text),clGreen);
end;
```

Setelah program dieksekusi maka akan muncul tampilan seperti berikut ini



B. Soal-soal

Buatlah sebuah program untuk menampilkan diagram batang pada data berikut ini

TAHUN JENIS PRODUKSI	**1993**	**1994**	**1995**
PADI	850	925	1250
JAGUNG	525	625	350
KEDELAI	725	700	875
KENTANG	350	425	865
KOPI	95	100	85
JUMLAH	1945	2575	3425

Bagian 1

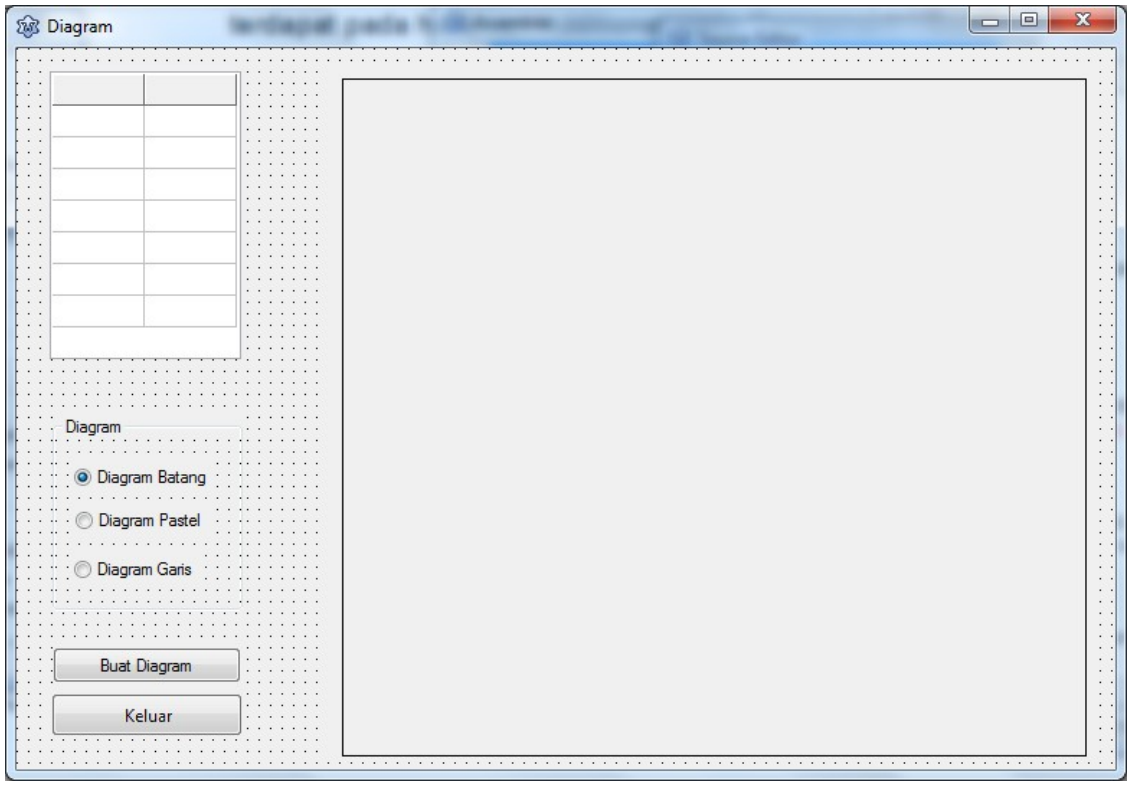
FUNGSI CHART

A. Chart

Komponen Chart berguna untuk membuat berbagai jenis diagram seperti Diagram Garis, Diagram Lingkaran dan Diagram batang dalam Lazarus. Dengan menggunakan komponen ini, penyajian grafik berdasarkan sekumpulan data dapat dilakukan dengan mudah. Komponen Chart terdapat pada halaman Additional pada Component Palette.

Latihan 1

Dalam latihan kali ini kita akan membuat berbagai diagram menggunakan fungsi Chart. Untuk itu buatlah form dengan tampilan sebagai berikut:



Kemudian atur propertiesnya sebagai berikut:

Komponen	Properties	Value
Button1	Caption	Buat Diagram
Button1	Caption	Keluar
Stringgrid1	Colcount	2
	RowCount	8
RadioButton1	Caption	Diagram Batang
RadioButton1	Caption	Diagram Pastel
RadioButton1	Caption	Diagram Garis
Chart		

Tambahkan beberapa hal didalam bagian awal unit

```
uses
  Classes, SysUtils, FileUtil, LResources, Forms, Controls, Graphics, Dialogs,
Grids, ExtCtrls, StdCtrls, TAGraph, TASeries, Buttons;


type
  { TForm1 }
  TForm1 = class(TForm)
    BitBtn1: TBitBtn;
    Button1: TButton;
    Chart1: TChart;
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    StringGrid1: TStringGrid;
    procedure BitBtn1Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
```

```
  b: TBarSeries;
  s: TSerie;
  p: TPieSeries;
 public
  { public declarations }
end;
```

Klik dua kali pada button1 dan masukkan kode sebagai berikut

```
procedure TForm1.Button1Click(Sender: TObject);
var x:array[1..7] of integer;
   banyak:array[1..7] of real;
   i,n:integer;
begin
   chart1.;
   chart1.BottomAxis.Visible:=TRUE;
   chart1.LeftAxis.Visible:=TRUE;
   n:=7;
   for i:=1 to n do
   begin
      x[i]:=strtoint(stringgrid1.cells[0,i]);
      banyak[i]:=strtofloat(stringgrid1.cells[1,i]);
   end;
   if radiobutton2.checked then
     begin
     p := TPieSeries.Create(chart1);
     Chart1.AddSerie(p);
     p.title := 'pie';
     p.SeriesColor := clRed;
```

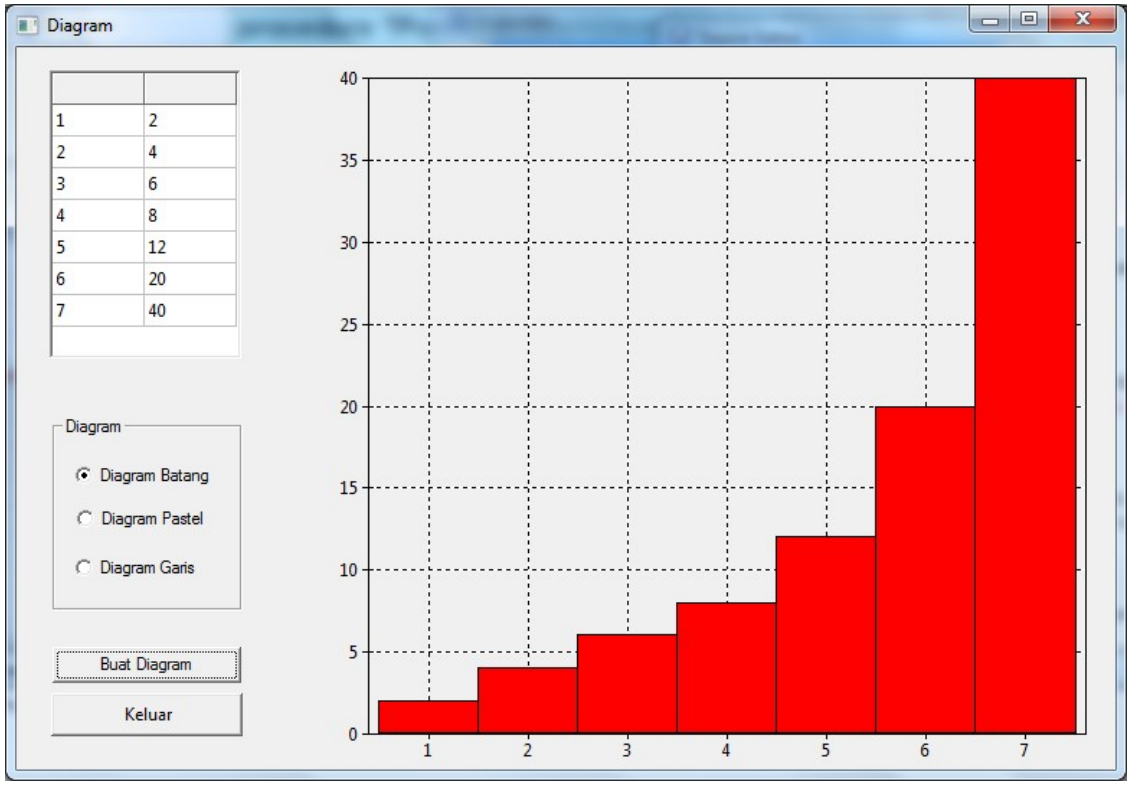
```
    for i:=1 to n do
      p.AddPie(banyak[i],inttostr(x[i]),clTAcolor);
    end
  else if radiobutton1.Checked then
    begin
    b := TBarSeries.Create(Chart1);
    Chart1.AddSerie(b);
    b.BarWidthPercent:=100;
    for i:=1 to n do
      b.AddXY(x[i],banyak[i],'',clred);
    end
  else
    begin
    s:= TSerie.Create(Chart1);
    s.ShowLines := true;
    s.ShowPoints := true;
    s.Pointer.Style := psRectangle;
    s.title := 'line';
    s.SeriesColor := clRed;
    Chart1.AddSerie(s);
    for i:=1 to n do
      s.AddXY(x[i],banyak[i],'',clred);
    end;
end;
```

Klik dua kali pada button2 dan masukkan kode sebagai berikut

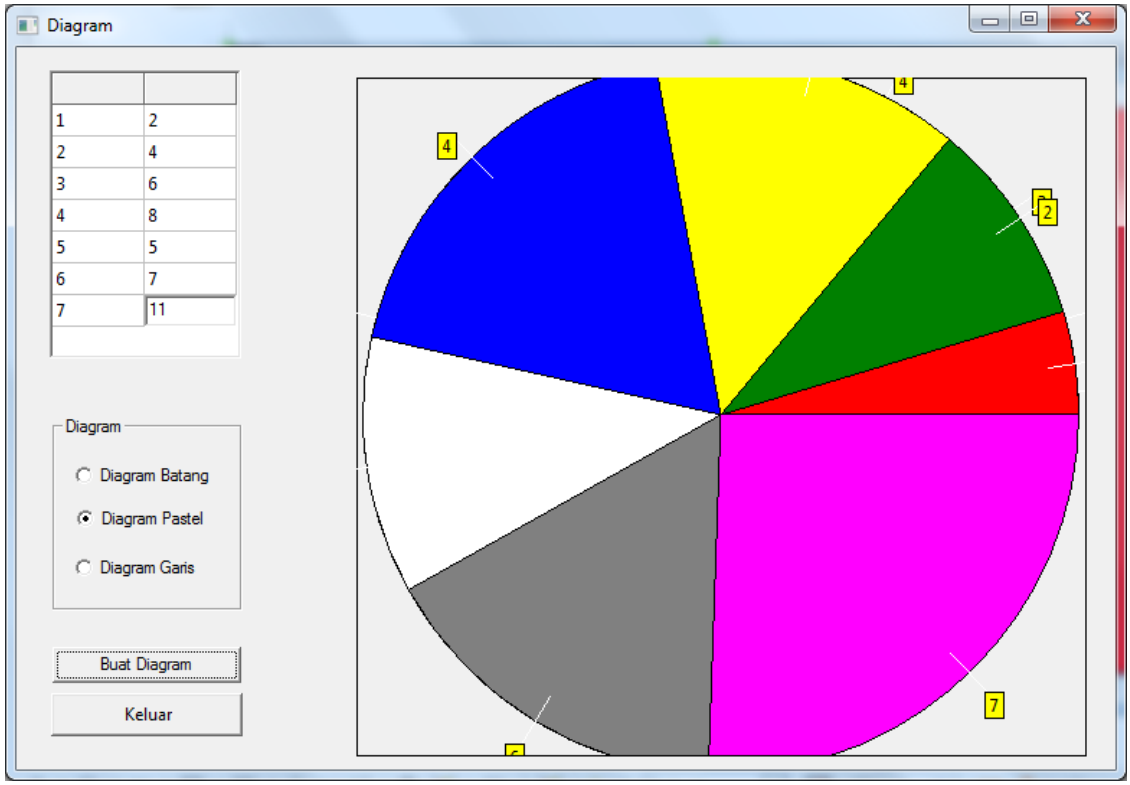
```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  close;
end;
```

Setelah program dieksekusi maka akan muncul tampilan seperti berikut ini

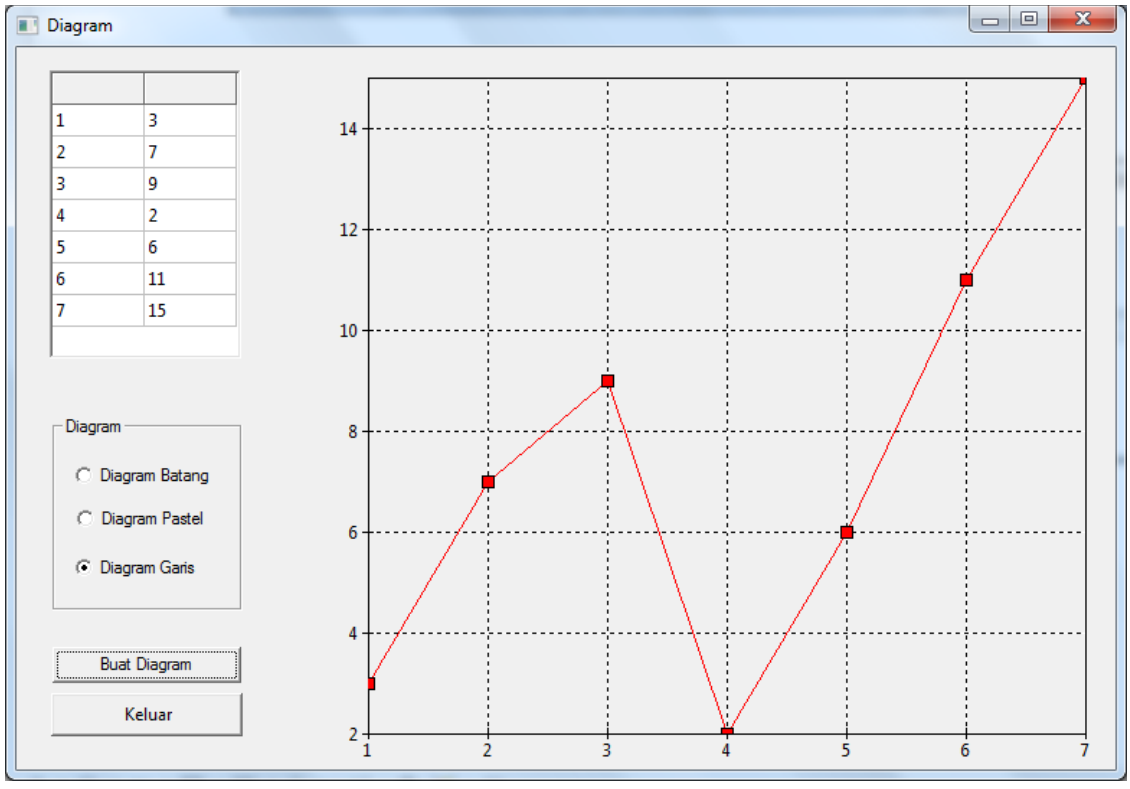
Untuk Diagram Batang



Untuk Diagram Lingkaran



Untuk Diagram Garis



B. Soal-soal

Buatlah sebuah program untuk menampilkan diagram lingkaran dan diagram batang pada data berikut ini

Sekolah	Banyak Siswa
SD	1562
SLTP	1091
SLTA	826
PT	235

MODUL 8

MENGAKSES BERKAS DAN PESAN KESALAHAN

PENDAHULUAN

Dalam modul ini akan dipelajari mengenai cara mengakses berkas melalui Lazarus dan menemukan dan memperbaiki kesalahan melalu pesan kesalahan atau **messages**.

Modul ini terdiri atas dua bagian yaitu pada bagian pertama mengenai mengakses berkas yang meliputi cara-cara membuat berkas, menyimpan berkas, dan membuka kembali berkas yang telah disimpan.

Pada bagian kedua dijelaskan mengenai cara menemukan kesalahan, menentukan jenis kesalahan dan memperbaiki program yang mengandung kesalahan. Didalam Lazarus apabila kita menulis suatu program dan ternyata dari susunan ataupun syntax yang digunakan tidak benar maka akan muncul pesan kesalahan (*error*) di jendela **Messages**. Dalam bagian ini kita akan gunakan pesan ini untuk memperbaiki program yang kita buat.

Bagian 1

MENGAKSES BERKAS

A. Mengakses Berkas

Data yang kita miliki selalu kita simpan dalam suatu berkas. Berbagai jenis berkas dipergunakan untuk data dengan kelebihan dan kekuranggannya masing-masing. Pemilihan jenis berkas biasanya dikaitkan dengan kecepatan akses, kemudahan dalam penyimpanan dan ukuran memori yang digunakan. Berkas dipergunakan untuk meyimpan data biasanya dalam bentuk file, dalam bahasa pemograman Lazarus berkas dibagi menjadi :

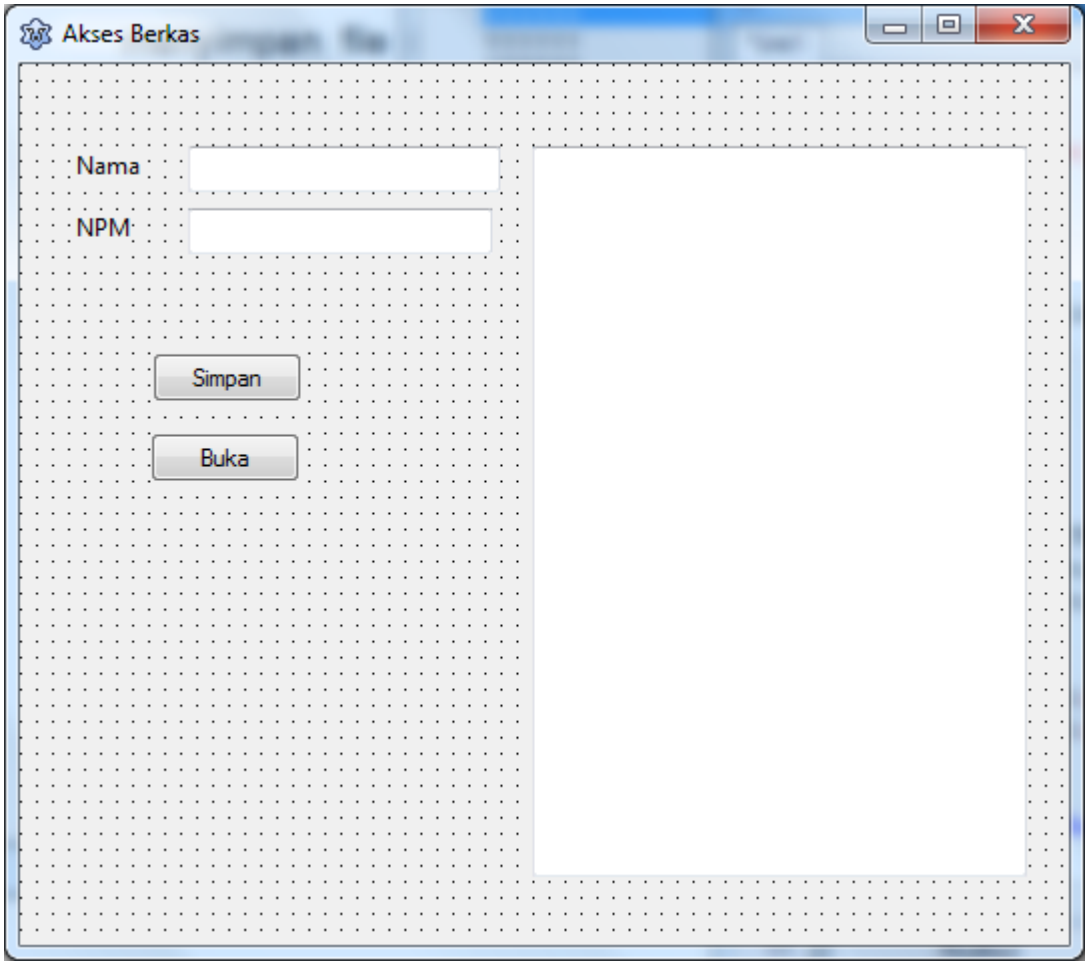
1. Berkas bertipe
2. Berkas teks
3. Berkas tak-bertipe

Berkas bertipe adalah berkas yang datanya memiliki tipe. Dengan menggunakan berkas bertipe kita bisa meyimpan data yang bertipe seperti real, integer dan bahkan yang berupa struktur data seperti *record*. Beberapa prosedur yang dipergunakan dalam meyimpan berkas bertipe adalah:

Nama Prosedur	**Kegunaan**
Prosedure AssignFile(var F;FileName:string)	Untuk mengaitkan variabel berkas dengan nama fisik berkas **Contoh** AssignFile(Varberkas,'ABC.dat');
Rewrite(VariabelBerkas)	Menciptakan berkas dan membukanya
Reset(VariabelBerkas)	Membuka berkas
CloseFile(VariabelBerkas)	Menutup berkas
Write(VariabelBerkas,VariabelData,...)	Menyimpan sebuah *record* data
Read(VariabelBerkas,	Membaca data

Latihan 1

Dalam latihan kali ini kita akan membuat sebuah program untuk membuka dan menyimpan file dalam bentuk teks. Buatlah Form dengan tampilan sebagai berikut:



Komponen	Properties	Value
Button1	Caption	&Simpan
Button2	Caption	&Buka
Edit1	Text	
Edit2	Text	
Label1	Caption	Nama
Label2	Caption	NPM
Memo1	Lines	

Pada form dibawah kata **type** ketiklah kode berikut ini:

```
TSiswa = Record
    Nama:string[30];
    NPM:string[30];
   end;

private
   { Private declarations }
   BerkasSiswa: file of TSiswa;
   Terbuka:Boolean;
```

Klik dua kali pada button1 dan masukkan kode sebagai berikut :

```
procedure TForm1.Button1Click(Sender: TObject);
var
NamaBerkas:string;
 DataSiswa:TSiswa;
begin
  NamaBerkas:='siswa.dat';
  assignFile(BerkasSiswa,NamaBerkas);
   if FileExists(NamaBerkas) then
    Reset(BerkasSiswa)
   else
    Rewrite(BerkasSiswa);
   seek(BerkasSiswa,FileSize(BerkasSiswa));
 With DataSiswa do
 begin
  Nama:=edit1.Text;
  NPM:=edit2.Text
 end;
```

```
 write(BerkasSiswa,DataSiswa);
 Edit1.Text:='';
 Edit2.Text:='';
 edit1.SetFocus;
end;
```

Klik dua kali pada button2 dan masukkan kode sebagai berikut :

```
procedure TForm1.Button2Click(Sender: TObject);
Var
 NamaBerkas:string;
 Data:TSiswa;
begin
 NamaBerkas:='siswa.dat';
 Terbuka:=false;
 if FileExists(NamaBerkas) then
 begin
  assignFile(BerkasSiswa,NamaBerkas);
  Reset(BerkasSiswa);
  Terbuka:=True;
 end
 else
 begin
   Messagedlg('File tidak ada',mtwarning,[mbok],0);
   close;
 end;
 memo1.Lines.Clear;
 while not Eof(BerkasSiswa) do
 begin
```

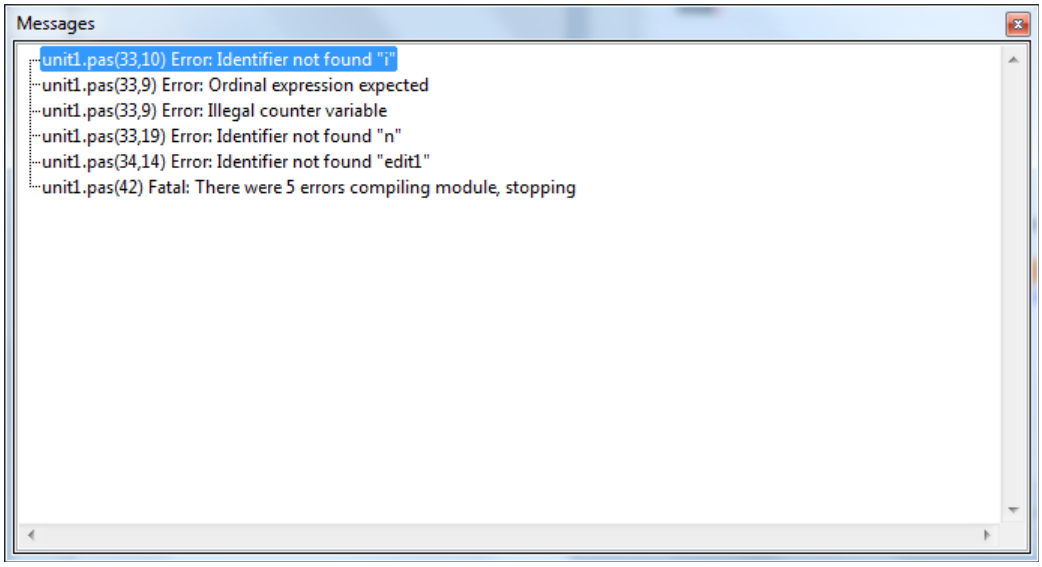
```
    read(BerkasSiswa,Data);
     memo1.Lines.Add(data.Nama);
     memo1.Lines.Add(data.NPM);
  end;
end;
```

Bagian 2

MENEMUKAN PESAN KESALAHAN

A. Pesan Kesalahan

Didalam Lazarus apabila kita menulis suatu program dan ternyata dari susunan ataupun syntax yang digunakan tidak benar maka akan muncul pesan kesalahan (*error*) di jendela **Messages**.



Selain pesan kesalahan dalam jendela **messages** juga diberikan warning apabila ada sesuatu pada program kita dan keterangan apabila program kita telah sukses. Pada gambar diatas terlihat bahwa ada 5 kesalahan yang terjadi yang di tunjukkan pada setiap baris untuk setiap kesalahan. Apabila kita klik pada baris tertentu dalam jendela messages maka secara otomatis dalam source editor menunjukkan baris yang berisi kesalaha yang dimaksud, sehingga kita bisa dengan mudah untuk memperbaikinya. Berikut akan dijelaskan mengenai beberapa pesan kesalahan yang sering muncul dalam Lazarus

1. Out Of Memory

Memory tak cukup untuk melakukan proses (misalnya mengkomplikasi) kalau sedang berada dalam IDE dianjurkan mengkomplikasi program ke disk.

2. Identifier Expected

Pengenal (variable) diperlukan pada posisi yang ditunjukkan kemungkinan disebabkan yang dipakai adalah reserved word (kata tercadang).

3. Unknow Identifier

Pengenal (variabel atau konstanta) belum dideklarasikan.

4. Duplikate Identifier

Ada pengenal dengan nama yang sama.

5. Sintax Error

Ada karakter yang tidak diperkenankan, biasanya disebabkan kekurangan tanda petik pada string.

6. Error In Real Constant

Salah dalam penulisan konstanta real.

7. Error In Integer Constant

Salah dalam penulisan konstanta integer.

8. String Constant Exred Line

Salah karena string belum diakhiri dengan tanda petik.

9. To Many Masted Files

File include yang memanggil file include perlu dikurangi.

10. Unexpeted End Of File

Biasanya disebabkan salah penulisan Begin dan End atau ada suatu komentar yang belum ditutup.

11. Line To Long

Panjang karakter dalam baris melebihi 126 karakter.

12. Type Identifier Expected

Pengenal type belum diberikan.

13. Too Many Open File

Terlalu banyak file yang dibuka (dapat diatur melalui file CONFIG.SYS).

14. Infalid File Name

Nama file salah.

15. File Not Found

File tidak ditemukan.

16. Disk Full

Disk penuh.

17. Infalid Compiler Directive

Pengarah computer yang ditunjukkan tidak dikenal.

18. Too Many Files

Terlalu banyak file yang dilibatkan sewaktu melibatkan komplikasi unit atau program.

19. Undifined Type In Pointer Definition

Type yang digunakan pada pendenifisian pointer belum dideklarasikan.

20. Variable Identifier Expected

Pengenal yang ditunjuk seharusnya menyatakan sebuah variable.

21. Error In Type

Kesalahan dalam pendefinisian type.

22. Structure Too Large

Type struktur terlalu besar (ukuran maksimal yang diperkenankan yaitu 65520 byte).

23. Set Base Type Out Of Range

Type dasar dari himpunan harus berbeda dalam kawasan 0 sampai 255 atau berupa type enumerasi yang jumlah kemungkinannya tidak lebih dari 256 buah.

24. File Compotitions My Not Be Files

Komponen dari file tidak boleh berupa file.

25. Invalid String Length

Panjang string dalam pendeklarasian haruslah terletak antara 1 sampai 155.

26. Type Mismach

Type tidak cocok.

27. Invalid Subrange Base Type

Kesalahan pada type dasar subrange.

28. Lower Bound Greater Than Upper Bound

Dalam mendeklarasikan subrange, nilai awal harus lebih kecil daripada nilai akhir.

29. Ordinal Type Expected

Type yang diperkenalkan adalah type original.

30. Integer Constant Expected

Mengharapkan suatu konstanta.

31. Constant Expected

Mengharapkan suatu konstanta.

32. Integer Or Real Constant Expected

Mengharapkan konstanta real/integer.

33. Type Identifier Expected

Mengharapkan pengenal type.

34. Infalid Function Result Type

Type keluaran fungsi salah (seharusnya berupa type sederhana, string / pointer).

35. Label Identifier Expected

Mengharapkan pengenalan label.

36. Begin Expected

Kurang Begin

37. End Expected

Kurang End.

38. Integer Expression Expected

Mengaharapkan ungkapan integer.

39. Ordinal Expression Expected

Mengharapkan ungkapan ordinal.

40. Boolean Expression Expected

Mengaharapkan ungkapan Boolean.

41. Operand Types Do Not Match Oprator

Type operand tidk sesuai dengan operator.

42. Error In Expression

Kesalahan dalam penulisan ungakapan .

43. Illegal Usignment

Kesalahan dalam pernyataan penugasan.

44. Field Identifier Expected

Mengaharapkan field dari record.

45. Object File Too Large

File object yang berukuran lebih dari 64 kb tidak dapat di link oleh turbo pascal.

46. Undefined External

Prosedur / fungsi eksternal belum di definisikan.

47. Invalid Object File Record

Ada suatu object record dalam file object yang tidak sah.

48. Code Segment Too Large

Kode dalam segment kode terlalu besar prosedur / fungsi perlu dipecah dalam beberapa unit.

49. Data Segment Too Large

Data dalam segmen data terlalu besar tempatkan data dalam heap.

50. Do Expected

Kurang Do.

51. Invalid Public Definition

Kesalahan yang berkaitan dengan bahasa assembly.

52. Invalid Extrn Definition

Kesalahan yang berkaitan dengan bahasa assembly.

53. Too Many Extrn Definition

Kesalahan yang berkaitan dengan bahasa assembly.

54. Of Expected

Kurang Of.

55. Interface Expected

Kurang Interface.

56. Invalid Relocatable Reference

57. Then Expected

Kurang Then.

58. To Or Down To Expected

Kurang To atau DownTo.

59. Underfined Forward

Definisi dari prosedur / fungsi belum diberikan.

60. Too Many Procedures

Terlalu banyak prosedur / fungsi jumlah fungsi.

61. Invalid TypeCast

Kesalahan dalam melakukan typecast (konversi type).

62. Division By Zero

Kesalahan karena pembagian bilangan dengan nol (0).

63. Invalid File Type

Type file yang digunakan tidak mengenal prosedur . fungsi penangan file yang ditunjuk (misalnya file teks tidak mengenal prosedur seek).

64. Cannot Read Or Write Variables Of This Type

Type dari variable yang ditunjuk tidak dapat dikenalkan pada instruksi Write dan Writeln atau Read dan Readln.

65. Pointer Variable Expected

Mengaharapkan variable pointer.

65. ';' Expected

Kurang titik koma pada akhir baris.

B. Soal-soal

1. Perhatikan program dibawah ini:

```
var i : byte;
  begin
        i:=0;
        while i > -10 do
          begin
            listbox1.items.add(i);
            dec(i);
        end;
   end;
```

Temukan dan perbaiki kesalahan yang ada!

2. Temukan dan perbaiki kesalahan yang ada pada program berikut ini!

```
Var i,a,r,n,S,Si:integer;
        Sn:real;
begin
        n:=strtoint(edit1.text);
        Si:=0;
        for i:=1 to n do Si:=Si+2*exp((i-1)*ln(3));
        S:=round(Si);
        edit2.text:=inttostr(S);
        Sn:=2*(1-exp(n*ln(3)))/(1-3);
        S:=round(Sn);
```

```
        edit3.Text:=inttostr(S);
end;
```

3. Telusuri bagian program berikut ini!

```
        var a,b,i,n:integer;
         jum1,jum2,hasil:real;
        begin
        a:=strtoint(edit4.text);
        b:=strtoint(edit5.text);
        n:=1000000;
        if a<0 then
        begin
         jum1:=0;jum2:=0;
        for i:=1 to n/2 do jum1:=jum1+(3*sqr(a-a*i/n)+2*(a-a*i/n)+4);
        for i:=1 to n/2 do jum2:=jum2+(3*sqr(b*i/n)+2*b*i/n+4);
        hasil:=-a*jum1/n+b*jum2/n;
        end;
        else
        begin
         jum1:=0;
         for i:=1 to n do jum1:=jum1+(3*sqr(a+(b-a)*i/n)+2*(a+(b-a)*i/n)
        +4);
        hasil:=(b-a)*jum1/n;
        end;
        edit6.Text:=inttostr(hasil);
        end;
```

Pada baris manakah bagian program yang salah (error)

MODUL 9

OBJECT ORIENTED PROGRAMMING

PENDAHULUAN

Dalam modul ini akan dipelajari mengenai pengertian Pemrograman berorientasi objek (Object Oriented Programming) atau biasa dsingkat OOP.

Modul ini terdiri atas dua bagian yaitu pada bagian pertama mengenai membuat unit yang terdiri atas pengertian unit dan kelas, menambahkan metode dan pengertian destruktor.

Hampir semua perangkat lunak (*Software*) pengembangan aplikasinya mengarah pada pemograman berorientasi objek. Hal ini berangkat dari kenyataan bahwa dengan menggunakan pendekatan berorientasi objek, suatu kode dapat digunakan kembali pada sejumlah program dengan mudah dalam pemograman hal ini disebut *reusability*.

Ada tiga karakteristik yang mendasari pemograman berorientasi objek :

a) Pengkapsulan : Pengemasan antara data dan prosedur atau fungsi dalam satu wadah yang disebut objek.

b) Pewarisan : Kemampuan suatu objek untuk menurunkan karakteristik yang dimilikinya

c) Polimorfisme : Suatu sifat yang memungkinkan nama yang sama dapat menyatakan tindakan yang berbeda.

Pada bagian 2 akan dibahas tentang pembuatan software statistika yang melibatkan semua kaidah-kaidah dalam pemrograman Lazarus, yang telah dijelaskan pada bab-bab sebelumnya. Pembuatan software statistika yang dijelaskan pada modul ini dibatasi pada fungsi-fungsi statistika dasar yang meliputi ukuran pemusatan, ukuran letak dan ukuran variasi.

Bagian 1

MEMBUAT UNIT

A. Objek dan Kelas

Kelas pada object pascal tidak lain adalah suatu jenis tipe data terstruktur. Kelas mendefinisikan suatu struktur yang tersiri atas field, metode dan properti

a) Field

   Variabel yang merupakan bagian dari kelas, mirip ***field*** pada struktur ***record***

b) Metode

   Subrutin (dapat berupa fungsi atau prosedur)

c) Properti

   Field yang memiliki penentu akses (berupa read dan write)

Suatu kelas dideklarasikan dengan bentuk sebagai berikut:

```
Type
  Namakelas=class(kelasOrangtua)
    Daftar Anggota
End;
```

Contoh:

```
Type
  TStatdasar = Class
```

```
    Rata2   :real;
    Varians :real;
End;
```

B. Menambah Metode

Field-field yang berada pada suatu kelas umumnya diakses melalui metode. Meode pada objek biasanya berupa fungsi dan prosedur atau konstruktor. Konstruktor sendiri adalah metode khusu yang menciptakan dan memberi nilai awal objek. Konstruktor diciptakan denga perintah:

```
Constructor Create;
Constructor Create(parameter);
```

Latihan 1

Buatlah sebuah unit dalam program dibuat dengan cara memilih ***File > New > Unit.*** Save lah unit yang sudah dibuat dengan nama **Statdasar** lalu tuliskan kode berikut ini:

```
unit Stats;

interface
type
 TStats = class
 function KV(S:real;X:real):real;
end;

implementation
```

```
function TStats.KV(S:Real;X:Real):real;
begin
 KV:=S/X*100;
end;
end.
```

Setelah suatu kelas dideklarasikan, kelas dapat digunakan untuk mendeklarasikan variabel pada bagian Var contoh:

Var

```
 hasil:TStats;
```

Pada contoh di atas, hasil merupakan variabel yang menunjuk ke objek yang berkelas **TStatdasar**. Pendeklarasian di atas tidak menciptakan objek, untuk menciptakan objek, di pergunakan perintah sebagai berikut:

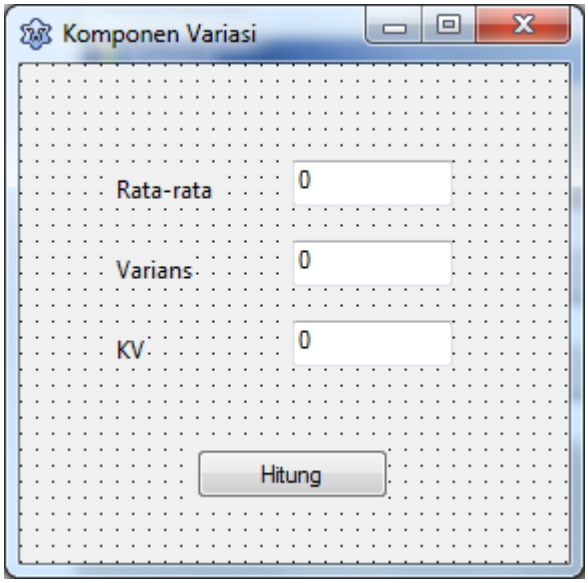
```
Variabel:=Namakelas.NamaKonstruktur (Parameter);
```

Contoh

```
hasil:=TStats.create(rata,varians);
```

Latihan 2

Pada latihan berikut kita akan membuat program untuk menghitung rata-rata, varians dan kompenen variasi (KV) menggunakan OOP. Untuk itu buatlah sebuah form dengan tampilan sebagai berikut:



Kemudian atur propertiesnya sebagai berikut

Komponen	Properties	Value
Edit1	Text	0
Edit2	Text	0
Edit3	Text	0
Button1	Caption	&Hitung

Pada bagian awal program tambahkan dalam uses unit **Stats**.

uses

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

Ketikan Statdasar untuk memberitahu bahwa program yang dibuat menggunakan **unit Stats**

Klik dua kali pada button1 dan isikan kode sebagai berikut:

```
procedure TForm1.Button1Click(Sender: TObject);
Var
 hasil:TStatdasar;
 rata,varians:real;
begin
 rata:=strtofloat(edit1.Text);
 varians:=strtofloat(edit2.Text);
 hasil:=TStats.create;
 edit3.Text:=floattostr(hasil.KV(rata,varians));
end;
end.
```

C. Destruktor

Destruktor adalah metode khusu pada kelas yang berguna untuk melakukan pembebasan memori (bila objek tidak dipakai lagi). Bentuk metode ini pada prinsipnya serupa dengan konstruktor, dengan kata **destructor**.

Destructor Destroy

Contoh:

Hasil.destroy;

D. Visibilitas

Setiap anggota kelas memiliki visibilitas. Visibilitas menentukan bisa tidaknya suatu anggota kelas diakses di unit lain atau dalam suatu program. Contohnya

Hasil.Rata2 :=2;

Hasil.Varians :=1;

Hal diatas menunjukkan bahwa visibilitas kedua ***field*** tersebut berdifat ***public*** (dapat diakses dari mana saja). Visibilitas anggota kelas (field maupun metode ditentukan melalui salah satu diantara lima kata tercadang sebagai berikut:

a) Private

   Anggota kelas tidak bisa diakses di luar unit tempat anggota dideklarasikan.

b) Protected

   Anggota kelas dapat diakses pada modul tempat kelas dideklarasikan serta pada kelas turunannya.

c) Public

   Anggota kelas dapat diakses dari mana saja.

Latihan 3

Ubahlah kode pada Latihan 1 menjadi sebagai berikut:

```
unit Stats;
interface
type
 TStats = class
 protected
 function KV(S:real;X:real):real;
end;
implementation
function TStats.KV(S:Real;X:Real):real;
begin
```

```
 KV:=S/X*100;
end;
end.
```

Apakah program masih dapat berjalan ?

E. Pewarisan

Pewarisan menunjukkan suatu kelas menurunkan karakteristik yang dimiliki ke kelas lain. Berkaitan dengan hal ini terdapat istilah kelas ***descendant*** dan kelas ***ancestor***.

- Kelas ***ancestor*** adalah kelas yang mewariskan karakteristik ke kelas lain.
- Kelas ***descendant*** adalah kelas yang mewariskan karakteristik ke kelas lain.

Latihan 4

Tambahkanlah kode dibawah ***implementation*** pada unit stats sebagai berikut:

```
type
 TMyStats=class(TStats)  //membuat klas turunan
end;
```

F. Soal-soal

1. Buatlah sebuah fungsi pada unit Stats untuk menghitung koefisien kemiringan dengan rumus sebagai berikut:

   Koefisien Kemiringan $= \frac{\bar{X} - Mo}{S}$

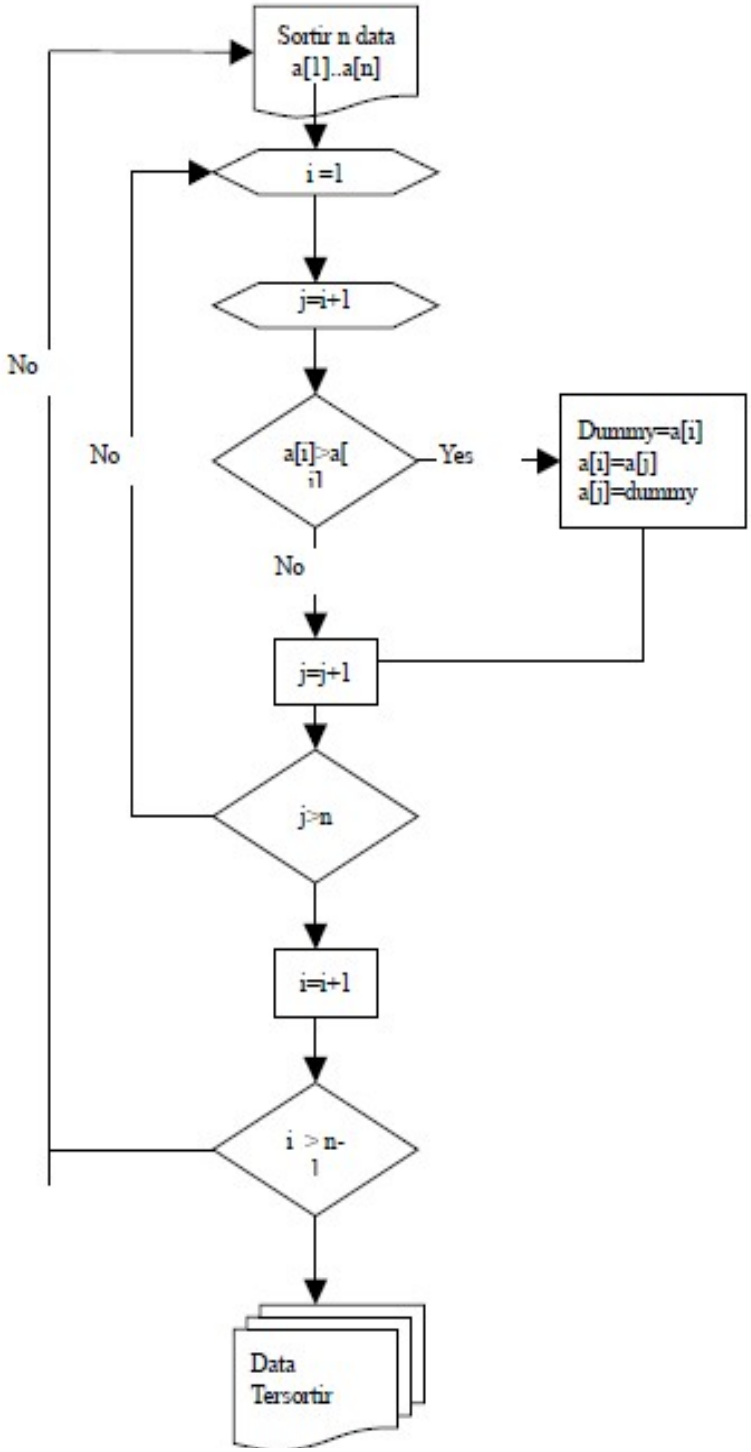
Dengan

$\bar{X}$ =Rata-rata

Mo = Modus

S = Standar Deviasi

2. Buatlah sebuah kelas dengan nama TMyStat yang merupakan turunan dari kelas TStats dan memiliki fungsi menghitung rata-rata ukur dari tiga buah bilangan sebut saja bilangan tersebut X_1 , X_2 dan X_3 ?
3. Buatlah suatu fungsi pada TMyStat untuk menghitung rata-rata hitung dari data pengamatan X_1 , X_2 ,..., X_n ?
4. Buatlah sebuah program dengan algoritma (*Flow chart*) untuk mengurutkan data dengan cara bubble sort sebagai berikut:


Sortir n data
a[1]..a[n]
i =1
j=i+1
No
No
a[i]>a[
i]
Yes
Dummy=a[i]
a[i]=a[j]
a[j]=dummy
No
j=j+1
j>n
i=i+1
i > n-
1
Data
Tersortir

Bagian 2

MEMBUAT SOFTWARE STATISTIKA

Dalam bagian ini kita akan membuat software statistika yang melibatkan semua kaidah-kaidah dalam pemrograman Lazarus, dan memanfaatkan potongan-potongan program yang telah dijelaskan pada modul-modul sebelumnya. Pembuatan software statistika yang dijelaskan pada bagian ini dibatasi pada fungsi-fungsi statistika dasar yang meliputi ukuran pemusatan, ukuran letak dan ukuran variasi.

A. Membuat Kelas

Sebagai langkah awal akan dibuat sebuah kelas dengan nama ***TMyStats*** pada unit **My*Stats*** sebagai berikut:

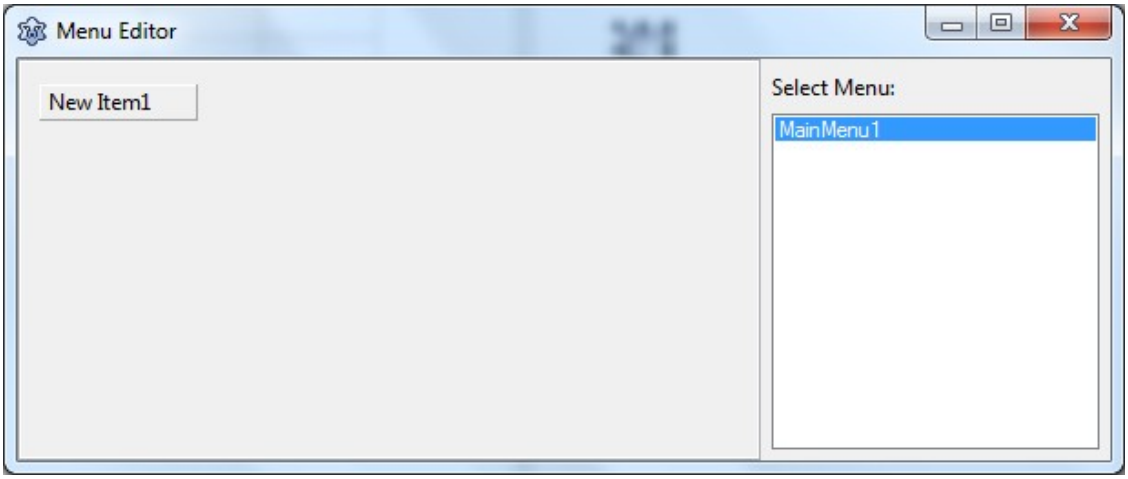
```
unit MyStats;


interface
type
 TMyStats = Class
 function ValMeans(X:array of real;n:integer):real;
 function ValVariance(X:array of real;n:integer):real;
end;
implementation
function TMyStats.ValMeans(X:array of real;n:integer):real;
var
 total:real;
 i:integer;
```

```
begin
  total:=0;
  for i:=1 to n do
  begin
  total:=total+x[i-1];
  end;
  Valmeans:=total/n;
end;
function TMyStats.ValVariance(X:array of real;n:integer):real;
var
 total:real;
 i:integer;
begin
 total:=0;
  for i:=1 to n do
  begin
  total:=total+(x[i-1]*x[i-1]);
  end;
  ValVariance:=(total-n*(ValMeans(x,n)*ValMeans(x,n)))/(n-1);
end;
end.
```

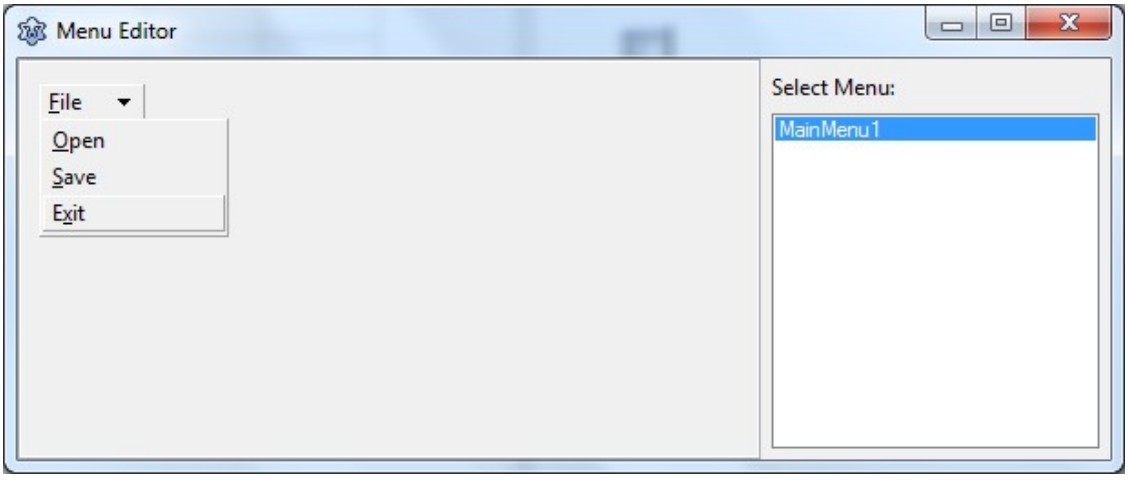
B. Membuat Menu pada Form

Untuk memudahkan dalam pengoperasian sebuah aplikasi, kita dapat menggunakan menu yang umumnya berada pada bagian atas Form. Didalam Lazarus kita bisa menambahkan menu dengan cara sebagai berikut

1. Buatlah sebuah Form
2. Tambahkan objek **MainMenu** pada Form, tempatkan sembarang.
3. Pada jendela Properties untuk MainMenu pilih **Items** maka akan muncul jendela berikut



4. Ubah Name menjadi 'MenuFIle' dan Caption menjadi 'File', jika ingin menambahkan huruf shortcut tambahkan Caption dengan tanda '&' menjadi '&File'
5. Klik kanan pada menu File dan pilih submenu, ubah properties Name menjadi 'MenuOpen' dan Caption menjadi '&Open' pada submenu menjadi
6. Tambahkan submenu lainnya dan ubah properties Name menjadi 'MenuSave' dan Caption menjadi '&Save'
7. Tambahkan submenu lainnya dan ubah properties Name menjadi 'MenuExit' dan Caption menjadi 'E&xit'. Maka jendela menu kita akan menjadi seperti ini

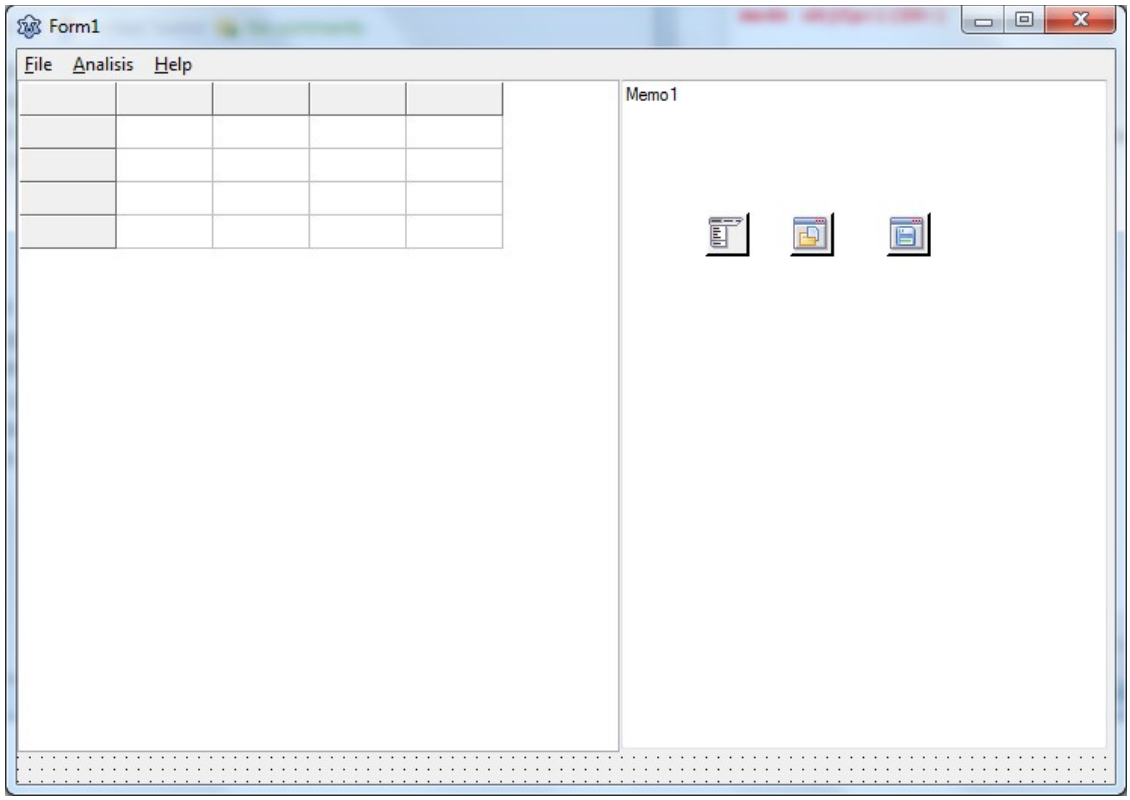


C. Mendesain Form

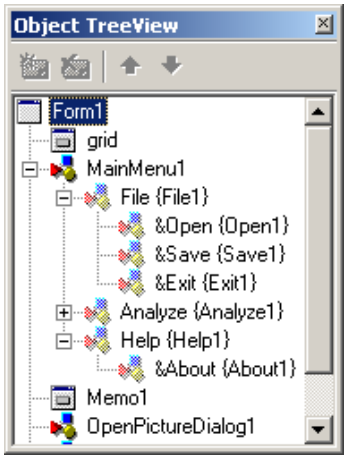
Pada Form1 yang telah kita buat tadi tambahkan beberapa objek yaitu **StringGrid**, **Memo**, **StatusBar**, **SaveDialog** dan **OpenDialog**, kemudian aturlah mengatur properties komponen yang dipergunakan sebagai berikut:

Komponen	Properties	Value
Stringgrid	Name	Grid
Memo	Name	Memo1
SaveDialog	Name	SaveDialog1
OpenDialog	Name	OpenDialog1
MainMenu	Name Items	MainMenu1 Open;Save;Exit Analyze;Descriptive Statistics Help;About

Aturlah Objek pada Form1 sehingga menjadi seperti berikut ini



Pindahkan ***Object Treeview*** pada Form1 sebagai berikut:



Pada Tab Events pada ***Object inspector*** untuk event OnCreate isikan kode sebagai berikut:

```
procedure TForm1.FormCreate(Sender: TObject);
var
 i,j:integer;
begin
 for i:=1 to form1.grid.RowCount do
   begin
     form1.grid.Cells[0,i]:=inttostr(i);
   end;
  for j:=1 to form1.grid.ColCount do
   begin
      form1.grid.Cells[j,0]:='Var'+inttostr(j);
   end;
end;
```

Tulislah dua buah prosedur dibawah ini dibawah implementation sebagai berikut:

```
procedure SaveStringGrid(StringGrid: TStringGrid; const FileName:
TFileName);
var
  f:    TextFile;
  i, k: Integer;
begin
  AssignFile(f, FileName);
  Rewrite(f);
  with StringGrid do
  begin
    // Write number of Columns/Rows
    Writeln(f, ColCount);
    Writeln(f, RowCount);
    // loop through cells
    for i := 0 to ColCount - 1 do
      for k := 0 to RowCount - 1 do
        Writeln(F, Cells[i, k]);
  end;
  CloseFile(F);
end;


// Load a TStringGrid from a file


procedure LoadStringGrid(StringGrid: TStringGrid; const FileName:
TFileName);
var
  f:         TextFile;
```

```
 iTmp, i, k: Integer;
 strTemp:    String;
begin
 AssignFile(f, FileName);
 Reset(f);
 with StringGrid do
 begin
  // Get number of columns
  Readln(f, iTmp);
  ColCount := iTmp;
  // Get number of rows
  Readln(f, iTmp);
  RowCount := iTmp;
  // loop through cells & fill in values
  for i := 0 to ColCount - 1 do
    for k := 0 to RowCount - 1 do
    begin
      Readln(f, strTemp);
      Cells[i, k] := strTemp;
    end;
 end;
 CloseFile(f);
end;
```

Klik satu kali pada menu File kemudian pilih Open dan tuliskan kode sebagai berikut:

```
procedure TForm1.Open1Click(Sender: TObject);
var
 FName:String;
begin
if opendialog1.Execute then
begin
 FName:=Opendialog1.FileName;
 loadstringgrid(grid,FName);
end;
end;
```

Klik satu kali pada menu Save dan isikan kode sebagai berikut:

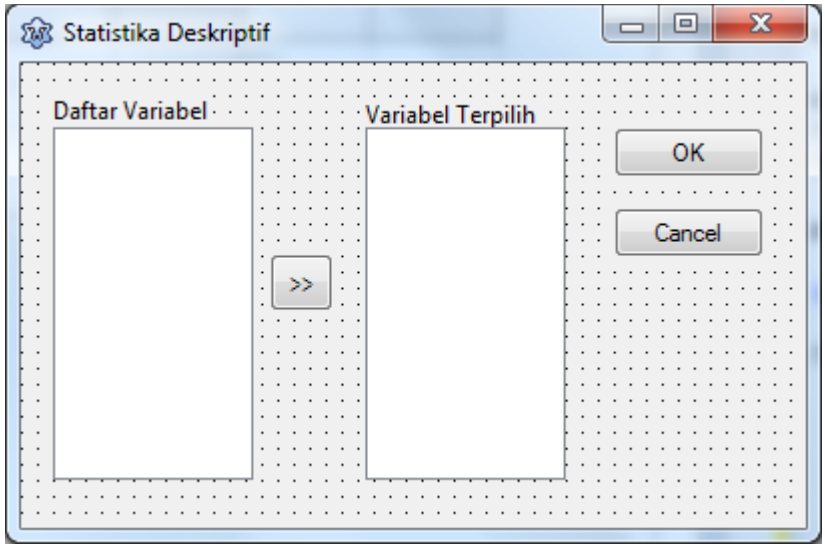
```
procedure TForm1.Save1Click(Sender: TObject);
Var
 FName:String;
begin
 if savedialog1.Execute then
 begin
   FName:=savedialog1.FileName+'.txt';
   savestringgrid(grid,FName);
 end;
end;
```

Klik satu kali pada menu Exit dan isikan kode sebagai berikut:

```
procedure TForm1.Exit1Click(Sender: TObject);
begin
   close;
end;
```

D. Form Statistika Deskriptif

Pada bagian sebelumnya kita telah memiliki sebuah Form yang telah berisi berbagai macam objek, akan tetapi untuk melakukan analisis kita memerlukan tambahan Form. Buatlah sebuah form baru dengan tampilan sebagai berikut:



Kemudian atur properties sebagai berikut

Komponen	Properties	Value
Listbox	Name	Varlist
Listbox	Name	Listbox1
Speedbutton	Caption	>>
Button	Name	Ok
Button	Caption	Cancel

Pada event ***TForm2.FormShow*** tuliskan kode sebagai berikut

```
procedure TForm2.FormShow(Sender: TObject);
var
  i,kolom : integer;
begin
  VarList.Clear;
  ListBox1.Clear;
  kolom:=form1.grid.ColCount-1;
  for i := 1 to kolom do
     VarList.Items.Add(form1.Grid.Cells[i,0]);
end;
```

Klik dua kali pada tombol ***SpeedButton1*** pada Form2 dan isikan kode sebagai berikut:

```
procedure TForm2.SpeedButton1Click(Sender: TObject);
var
  index, i : integer;
begin
   index := VarList.Items.Count;
   i := 0;
   while i < index do
   begin
      if (VarList.Selected[i]) then
      begin
        ListBox1.Items.Add(VarList.Items.Strings[i]);
        VarList.Items.Delete(i);
        index := index - 1;
```

```
          i := 0;
        end
        else i := i + 1;
    end;
end;
```

Klik dua kali pada tombol ***Button1*** pada Form2 dan isikan kode sebagai berikut:

```
procedure TForm2.Button1Click(Sender: TObject);
var
 i,j,noselected,kode:integer;
 Selected:array of integer;
 cellstring :string;
 X:array of real;
 rata,variasi:real;
 hasil:TMyStats;
begin
   noselected := ListBox1.Items.Count;
   SetLength(Selected,noselected);
   // Get selected variables
   for i := 1 to noselected do
   begin
      cellstring := ListBox1.Items.Strings[i-1];
      for j := 1 to 100 do
 if cellstring = form1.Grid.Cells[j,0] then selected[i-1] := j;
   end;
   Setlength(X,form1.grid.RowCount-1);
    form1.Memo1.Clear;
```

```
 for j:=1 to noselected do
  begin
   for i:=1 to form1.grid.RowCount-1 do
     begin
       x[i-1]:=strtofloat(form1.grid.cells[selected[j-1],i]);
     end;
     hasil:=TMyStats.Create;
     rata:=hasil.ValMeans(X,form1.grid.RowCount-1);
     variasi:=hasil.ValVariance(X,form1.grid.RowCount-1);
     form1.Memo1.Lines.Add('----------------------------');
     form1.Memo1.Lines.Add('Descriptive Statisics');
     form1.Memo1.Lines.Add('----------------------------');
              form1.Memo1.Lines.Add('Rata-rata              Var'+inttostr(j)+'=
'+floattostr(rata));
form1.Memo1.Lines.Add('Varians          Var'+inttostr(j)+          '=          '
+floattostr(variasi));
     form1.Memo1.Lines.Add('                        ');
     hasil.Destroy;
```

Kemudian eksekusi programnya dan apabila ada kesalahan silahkan diperbaiki dan di eksekusi kembali. Dari hasil program yang kita jalankan kita dapat melihat kekurangan dari program kita sehingga dapat diperbaiki untuk menjadi program yang lebih baik.

E. Soal-soal

Pada Program yang telah dibuat pada modul ini hanya menyertakan sebuah analisis yaitu Statistika Deskriptif. Modifikasi program tersebut dengan menambahkan submenu pada menu analisis yaitu

1. Diagram

dengan submenu Diagram Batang, Diagram Lingkaran. Kemudian buat isikan kode-kodenya agar bisa digunakan.

2. Pengujian Hipotesis Rata-rata

dengan submenu Satu sampel Z, Satu sampel t. Kemudian buat isikan kode-kodenya agar bisa digunakan.

DAFTAR PUSTAKA

Lukito, Ediman (1993) Belajar Sendiri Pemrograman dengan Turbo Pascal 7.0, PT Elex Media Komputindo, Jakarta.

Abel Azeem, Motaz (2012) Start Programming Using Object Pascal: Free Pascal/Lazarus Book. E-book by Motaz Abel Azeem, code.sd 30-08-2012.

http://id.wikipedia.org/wiki/Lazarus_%28perangkat_lunak%29

http://omophorest.blogspot.com/2012/04/pesan-error-di-pascal-dan-c.html

Buku Ini Diterbitkan Dalam Rangkaian
Program Pembuatan Buku Materi Ajar dan Kuliah Online
Universitas Padjadjaran 2013.