ABSTRACT

Dokumen ini merupakan bentuk format standar pengkodean Delphi/Kylix yang digunakan di lingkungan KIOSS Project. Beberapa diantaranya dibentuk berdasarkan konvensi dari Team Pengembang Kylix/Delphi.

Kami mengharap respon dari rekan-rekan sebagai koreksi, kritik dan saran. Kirimkan respon anda kepada Luri Darmawan di luridarmawan@kioss.com., atau melalui situs offisial kami di http://www.kioss.com

First Release
Semarang, 29 Juli 2002

Team KIOSS Project

Modifikasi Terakhir : 15 Februari 2003

DAFTAR ISI

CTRL+ALT+DEL

Dokumen ini memberikan aturan format standar dalam penulisan kode Kylix/Delphi Object Pascal di dalam lingkungan KIOSS Project. Aturan di dalamnya sebagian besar didasarkan pada konvensi internasional team pengembang Kylix/Delphi. Aturan main penulisan kode ini kami sebut sebagai **KIOSS Style**.

Latar Belakang

Kami sadari tiap kelompok mempunyai aturan sendiri dalam pengembangan aplikasi, tapi setidaknya aturan ini tidak melenceng jauh dari aturan yang telah ditetapkan oleh Borland sendiri. Tapi dalam KIOSS Project, kami mengharapkan kepada para rekan-rekan untuk mengubah gaya penulisan ke dalam KIOSS Style sebelum men-submit kan kode anda kepada KIOSS Project. Kami tidak memaksa Anda untuk mengubah konvensi/aturan Anda, tapi di dalam semua kode yang dibuat oleh KIOSS menggunakan aturan KIOSS Style ini. Kami benar-benar mengharapkan anda mengikuti aturan ini, saat anda mensubmitkan kode ke lingkungan KIOSS.

Jika para pengembang menggunakan aturan ini, akan memudahkan para pengembang sendiri dalam mempelajari semua kode yang ada di dalam KIOSS, dan juga akan menambah level kode yang dibuat pengembang, termasuk dalam *maintenance* dan *debugging*.

Tapi tidak selamanya konvensi ini menjadi dasar yang kaku dalam semua pengembangan. Kami juga tidak menganggap bahwa kami benar dan mereka salah, TIDAK. Tapi kami mempercayai, bahwa dengan adanya standarisasi ini, akan memberikan efisiensi yang lebih dalam pengembangan kami. Dengan adanya standariasi akan memudahkan dalam pembacaan kode, yang juga mungkinkan untuk dibaca oleh sekian banyak orang yang tergabung dalam KIOSS Project.
Pun apabila memang diinginkan, Anda boleh tetap mempertahankan format Anda sendiri, tapi apabila akan disubmitkan kepada KIOSS, Anda mesti mengkonversinya terlebih dulu dalam KIOSS Style.

Sebelum menutup antar muka ini, kami ucapkan terima kasih kepada para rekan-rekan pengembang, karena bergabung dengan KIOSS Project.

Kami mengharap respon dari rekan-rekan sebagai koreksi, kritik dan saran. Kirimkan respon anda kepada Luri Darmawan di luridarmawan@kioss.com.

KIOSS Project
http://www.kioss.com

Acknowledgments

Dokumen ini merupakan standar untuk kemudahan dan perawatan KIOSS Project, *user-supported library* dari proyek yang di desain dengan menggunakan Delphi ataupun Kylix. Segala kritik dan saran untuk pengembangan dokumentasi ini dapat disampaikan melalui milis KIOSS Project di http://groups.yahoo.com/group/kioss atau kioss@yahoogroups.com.
Untuk pedoman umum, dapat dibaca Delphi Style Guide, yang dipublikasikan oleh Charlie Calvert di http://www.borland.com/techvoyage/articles/DelphiStyle/StyleGuide.html .

Dokumen ini digunakan untuk semua pengerjaan dilingkungan KIOSS Project, keterangan lebih lanjut dapat menghubungi:

KIOSS Project Official Site
http://www.kioss.com
luridarmawan@kioss.com

Luri Darmawan
d.a. Gandaria Square
Jl. H. Syafei Hadzami Kav. 1 no. 1
Terusan Gandaria – Kebayoran Lama
Jakarta Selatan – 12240
Tel. 021 72796574
Fax. 021 72789222
ICQ# 175646805

Agus. eS
d.a. METAKOM
Jl. Alun-alun Selatan 10
Kauman – Semarang
Tel. 024 3585640
Fax. 024 3585641

Referensi

Dokumen ini menggunakan referensi dari sekian banyak sumber yang kami dapat dalam versi cetak ataupun dalam versi *e-book* yang sebagian besar kami dapatkan dari beberapa situs di internet.

Trademarks

KODE SUMBER

Setiap Kode Object Pascal dibagi dalam file Unit dan Proyek, yang keduanya menggunakan konvensi yang sama. File proyek Delphi memiliki ekstensi DPR, yang merupakan kode utama dari suatu proyek. Dan unit yang lain di dalam proyek tersebut menggunakan ekstensi PAS. Beberapa file tambahan, seperti html ataupun DLL, mungkin juga memainkan peranan di dalam proyek, dokumen ini hanya menggambarkan aturan dalam file DPR dan PAS saja.

Penamaan File Kode

Secara default memberikan penamaan dalam format 8:3, meskipun dimungkinan dengan penamaan yang mendukung *long file names*.

Penamaan nama unit harus dimulai dengan huruf "u". dan nama proyek (DPR) mengikuti dari nama proyek pengerjaan itu sendiri, serta unit/modul utama harus diawali dengan huruf "uz". Kecuali untuk file pustaka aplikasi, nama harus diawali dgn "pustaka"
Misal, suatu proyek pengembangan aplikasi stok yang dibuat dengan membagi kedalam beberapa modul barang, mutasi dan pelaporan, serta satu modul menu utama. Maka akan terbentu beberapa yang diantaranya (dalam urutan abjad) :

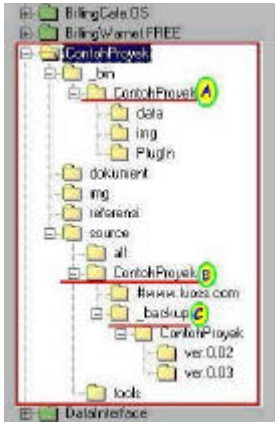
stok.dpr
stok.res
pustakaStok.pas
uBarang.pas
uMutasi.pas
uPelaporan.pas
uxPlugIn.pas
uyKonfigurasi.pas
uzMenuUtama.pas
…

Perhatikan bahwa setiap Aplikasi dibuat dilingkungan KIOSS Project, harus terdiri dari UnitUtama (uzXXXXXXX.pas) dan UnitKonfigurasi (uyKonfigurasi.pas). Dan jika menggunakan *PlugIn*, harus memasukkan modul plugin sebagai uxPlugIn.pas. Tentunya dengan nama file form sesuai dengan nama unit-nya, spt uMutasi.dfm untuk uMutasi.pas

Pengorganisasian Direktori

Ada baiknya jika kita selalu membuat struktur direktori yang sama untuk setiap pembuatan proyek aplikasi.
Pembuatan direktori ini dapat dilakukan secara otomatis, jika Anda menggunakan fitur "**Wizard Pembuatan Proyek**" di dalam **KIOSS.Assistant**.

Contoh struktur yang dibentuk dan yang disarankan dapat dilihat pada visualisasi berikut:



A. Tempat file *executable* diletakkan, dalam hal ini **ContohProyek.EXE**, dan setiap *executable file* selalu memiliki direktori data, img dan PlugIn.
B. Tempat *source code* .pas diletakkan. Ddirektori **all** berisi file kode yang kemungkinan digunakan oleh proyek lain dalam MainProyek ContohProyek.
C. Direktori backup untuk proyek "ContohProyek". Dengan fitur **Lokal Backup**, secara otomatis akan mengkopikan semua resource yang ada di dalam folder proyek ke dalam direktori versi yang secara otomatis di-*generate* oleh **KIOSS.Assistant**.

	Subdir	Purpose	File Extension Examples
	SOURCE	Project source files (project source, forms, units, include, resources)	DPR, PAS, DFM, RES
	_BIN\..	Compiled units & executables	EXE, DCU, DLL
	_BIN\..\..	file-file yang digunakan ketika menjalankan aplikasi.	BMP, JPG, DLL, DAT
	DOKUMEN	Dokumentasi Teknis dan Administratif	TXT, DOC
	REFERENSI	Semua file yang digunakan untuk referensi pengerjaan proyek.	*.*
	IMG	File image mentah	PSD, BMP

Pengorganisasian File

Semua unit seharusnya memiliki beberapa elemen berikut:
Blok Keterangan/Hak Cipta
Nama modul/unit
Interface
KVCS (KIOSS Version Control System)
Implementasi
End penutup dan titik
satu baris kosong sebagai pemisah elemen.

sehingga setiap unit memiliki kriteria berikut:

```
{=============================================================================
  @Author       : KIOSS Project, 2003, Luri Darmawan
                  http://www.kioss.com
  @Modul        : uNamaUnit.pas
  @Dibuat       : 2003-02-07 12:19:52
  @Deskripsi    : <Tulis deskripsi proyek anda disini, kalo mau!!!>
  @Kopirait     : KIOSS Project
  @Versi        : 0.01.xx
  @Modifikasi   :
  @OpenIssues   :
  @Histori      : <dibawah ini daftar histori kamu!!!>
  @Bug          : <List dibawah ini daftar error kamu!!!>
 =============================================================================}
unit KIOSSAssistantExpert;
{$I kiossINCLUDE.INC}

interface
uses
   kiossLIB, KIOSSExpert, windows, …;

CONST
  _KVCS_= '<nomor versi akan disisipkan secara otomatis>';
  {$R Version.RES}

implementation

end.
```

Apabila dikehendaki element tambahan yang Anda anggap penting, bisa pula anda tambahkan sendiri, atau bisa juga bila memang tidak dikehendaki bisa dihilangkan, kecuali HakCipta dan nama unit.

Dengan kondisi seperti ini, dapat diketahui informasi mengenai unit yang sedang dibuat, termasuk HakCipta, deskripsi unit/modul, nomor versi, histori pengerjaan dan daftar error/bug.

Semua sisipan diatas dapat dilakukan dengan mudah apabila Anda menggunakan Aplikasi **KIOSS.Assistant** yang bisa didownload di http://www.kioss.com

Deklarasi Unit

Setiap file *sourcecode* seharusnya diawali dengan teks *unit*, dalam huruf kecil. Dan diikuti dengan nama unit, yang diawali dengan huruf "u" kecil, kecuali untuk unit utama, unit konfigurasi dan plugin yang masing-masing harus diawali dengan:

uz	Unit Utama	contoh: uzStokBarang
uy	Unit Konfigurasi	contoh: uyKonfigurasi
ux	Unit PlugIn	contoh: uxPlugIn

Deklarasi Uses

Di dalam masing-masing unit, deklarasi *uses* seharusnya diawali dengan teks *uses* dengan huruf kecil, dan diikuti nama-nama unit yang digunakan dengan penulisan mengukiti pola masing-masing unit tersebut. Dipisahkan dengan tanda koma (",") dan diakhiri dengan tanda dititik koma (";").

sebagai contoh:

```
uses
  kiossLIB,
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;
```

Deklarasi Interface/Class

Deklarasi kelas diawali dengan dua (2) spasi, diikuti dengan identifier yang diawali huruf "T". Identifier sebaiknya diawali dengan huruf besar, dan memiliki huruf besar juga di tiap kata yang mengikutinya.
Misal:

```
  TKelasKu
```

diikuti 1 spasi, satu sama dengan, satu spasi, dan teks *class*, semua dalam huruf kecil

```
  TKelasKu = class( TObject)
```

Scoping directives seharusnya dua spasi dari margin kiri, dan dideklarasikan dengan urutan:

```
  TKelasku = class(TObject)
  private
  protected
  public
  published
  end;
```

Data seharusnya dideklarasikan di dalam seksi *private*, dan diawali dengan huruf F, diikuti pengenal tipe variabel dalam huruf kecil. Semua deklarasi tipe diletakkan 4 (empat) spasi dari batas kiri.

```
  TKelasku = class(TObject)
  private
    FiDataKu                          : Integer;
    procedure SetDataKu(const Value: Integer);
  protected
  public
    sNama                             :string;
  published
    property DataKu:Integer read FiDataKu write SetDataKu;
  end;
```

Perhatikan penamaan variabel pada seksi *public*, beda dengan variabel pada seksi *private*.
Akan dijelaskan kemudian.

PENAMAAN

Kecuali untuk kata yang dicadangkan dan direktif, yang semuanya dalam huruf kecil. Semua identifikasi Pascal seharusnya menggunakan format **InfixCaps**, artinya: setiap huruf pertama dalam tiap kata harus huruf besar, dan lainnya adalah huruf kecil, kecuali untuk singkatan, misalnya:

```
KelasKu, NamaPerusahaan, NomorKwitansi, NomorNPWP
```

Jangan pernah menggunakan *underscore* ("_") untuk memisahkan penggalan kata.

*** Selalu gunakan format InfixCaps, kecuali untuk kondisi tertentu yang dijelaskan kemudian ***

Penamaan Form

Diawali dengan huruf "**f**", dalam huruf kecil. Baca juga seksi **Deklarasi Unit** pada bahasan sebelumnya.

Penamaan Class / Interface

Diawali dengan huruf "**T**", dalam huruf besar. Baca juga seksi **Deklarasi Kelas** pada bahasan sebelumnya.

Penamaan Field

Baca juga seksi **Deklarasi Kelas** pada bahasan sebelumnya.
Diawali dengan hufur "**F**", dalam huruf besar, diikuti satu huruf tipe variabel dalam huruf kecil.

```
FiDataKu                    : Integer;
FsNamaPerusahaan            : string;
FeJumlahHutang              : extended;
```

pengecualian untuk penamaan tipe enumerasi:

```
TBitBtnKind = (bkCustom, bkOK, bkCancel, bkHelp, bkYes, bkNo,
  bkClose, bkAbort, bkRetry, bkIgnore, bkAll);
```

dalam kasus ini, huruf bk disisipkan sebelum nama tiap elemen dari enumerasi ini, bk berarti *ButtonKind*.

Perlu dipertimbangkan juga, penamaan field dengan satu-karakter sebaiknya dihindari, kecuali untuk penggunaan temporer atau sebagai variabel looping.
Hindari nama variabel l ("el"), karena sulit dibedakan dengan variabel 1 ("satu")

Penamaan Metode

Gunakan format **InfixCaps**. Jangan pernah menggunakan *underscore*/garis bawah (_) sebagai pemisahkata.

```
// Penamaan yang BAIK:
  TampilanStatus, GambarLingkaran, TambahKompen

// Penamaan yang TIDAK BAIK:
  TombolMouse       // kata benda, tidak menjelaskan fungsi
  gambarLingkaran   // dimulai dengan huruf kecil
  tambah_komponen   // ada underscores

// fungsi berikut tidak jelas maksudnya,
// jika bermaksud menjalankan sever, sebagiknya StartSever
// atau jika menguji apakah server sedang jalan,
//   sebaiknya IsServerRunning atau ApakahServerAktif
  ServerRunning     // verb phrase, tidak imperatif
```

Suatu metod yang membaca atau mengisi suatu *property of class*, seharusnya ditulis GetProperty atau SetProperty, dimana Property adalah nama dari *property* tersebut.

Contoh:

```
GetHeight, SetHeight
```

Metod yang digunakan untuk menguji suatu property bertipe boolean, sebaiknya diawali dengan kata "Is" atau "Apakah"

Contoh:

```
IsVisible, IsConnected
```

atau

```
ApakahTerhubung, ApakahSelesai
```

Khusus untuk prosedur/fungsi yang bersifat public dan bukan merupakan moted dari class, sebaiknya ditambahi awalan *underscore*/garisbawah (_)
Contoh:

```
_SetRegionalID
```

Rutin Lokal

Semua rutin fungsi/prosedur lokal harus diindentasi sebanyak dua karakter terhadap rutin level di atasnya.

```
procedure ContohProsedur;
var
  I: Integer;

  procedure LokalProsedur;
  begin
    ...
  end;
begin
  ...
  LokalProcedur;
  ...
end;
```

Penamaan Variabel

Penamaan Variabel Lokal

Diawali dengan huruf l (el) yg berarti lokal, dan dikuti huruf pengenal tipe variabel
misal:

```
var
  lsNama                          : string;
  liDataKu                        : integer;
  lslDaftarNama                   : TStringList;
```

Penamaan Variabel Public

Mirip dengan variabel lokal tanpa awalan huruf l (el, dan dikuti huruf pengenal tipe variabel
misal:

```
var
  sNama                          : string;
  iDataKu                        : integer;
  slDaftarNama                   : TStringList;
```

Penamaan Parameter

Seperti halnya penamaan variabel public, tapi diawali dengan huruf “p”, dengan huruf kecil.

```
function NamaFungsi( psParameter1:string; piParameter2:integer):word;
```

Kata yang dicadangkan

Kata yang dicadangkan dan semua direktif sebaiknya menggunakan huruf kecil.

Deklarasi Tipe

Deklarasi tipe harus diawali dengan huruf “**T**”, dalam huruf besar.

Baca bahasa mengenai **deklarasi kelas** diatas.

Penamaan Konstanta

Semua konstanta diawali dengan huruf “c”, dalam huruf kecil. Diikuti tipe konstanta dan pengenal konstanta. Antara pengenal dan nama, dipisahkan dengan *underscore*.
Misal:

```
const
  ci_JumlahMaksimum               = 1;
  cs_NamaPerusahaan               = 'KIOSS Project';

  // file
  csf_Konfigurasi                 = 'konfigurasi.konfig';
  csf_Aplikasi                    = 'aplikasi.exe';
  csf_NotePad                     = 'notepad.exe';
  // file gambar
  csf_imgPanelAtas                = 'panel.atas.bmp';
  csf_imgPanelBawah               = 'panel.bawah.bmp';
```



```
  csf_imgMenuUtama                = 'menu.utama.bmp';
  csf_imgPanelBawahKanan          = 'panel.bawah.kanan.bmp';
  csf_imgKananBawah               = 'kanan.bawah.bmp';
  csf_imgSudutKanan               = 'sudut.kanan.bmp';

  //  direktori
  csd_Windows                     = 'c:\windows\';
  csd_Root                        = 'c:\';

  // tabel
  cstbl_Sales                     = 'sales';
  cstbl_Customer                  = 'customer';

  // error
  ce_Sukses                       = 0;
  ce_Error                        = 255;
```

beberapa variasi lain yang muncul dikemudian hari, akan didokumentasikan segera mungkin.

Penamaan Resource String

Penamaan *resource string* diawali dengan “res” dalam huruf kecil.
Contoh:

```
resourcestring
  //------------------------------------------------------------------ BARANG  ---
  resScriptBarang  = ''
    + CRLF + 'CREATE TABLE <namatable> ('
    + CRLF + ' ID             <autoinc>,'
    + CRLF + ' kode           varchar( 16) NOT NULL,'
    + CRLF + ' kodegroup      varchar( 16) DEFAULT "",'
    + CRLF + ' nama           varchar( 50) DEFAULT "",'
    + CRLF + ' jml            float DEFAULT 0,'
    + CRLF + ' tipe           varchar( 16) DEFAULT "",'
    + CRLF + ' merk           varchar( 16) DEFAULT "",'
    + CRLF + ' sn             varchar( 20) DEFAULT "",'
    + CRLF + ' hjual          float DEFAULT 0,'
    + CRLF + ' hbeli          float DEFAULT 0,'
    + CRLF + ' kodesupplier   varchar( 16) DEFAULT "",'
    + CRLF + ' ket            varchar( 40) DEFAULT "",'
    + CRLF + ')';
```

Semua *resource srtring* harus dideklarasikan didalam file unit yang terpisah, sebaiknya diletakkan dalam file pustaka (“pustakaAplikasi.pas”) dimana “Aplikasi” disesuaikan dengan nama aplikasi yang sedang dibuat.

KOMPONEN

KIOSS Project telah menyusun sekumpulan komponen yang sering kali digunakan dalam setiap pembuatan aplikasi. Sebagian diantaranya merupakan hasil karya para tukang di kioss, dan sebagian besar lagi merupakan saduran dari komponen yang sudah ada.

Kebijakan Penggunaan Komponen

Apabila diperlukan, pemrogram dapat meletakkan komponen tambahan yang diperlukan ke dalam aplikasi yang dibuat, dengan catatan: menyertakan kode dari komponen yang digunakan. Akan lebih baik apabila pemrogram hanya menggunakan komponen standar dan atau komponen dari KIOSS. Komponen KIOSS dipublikasikan melalui web yang telah ditentukan, khususnya di http://www.kioss.com.

Sangat dimungkinkan, para developer menggunakan bermacam komponen yang ternyata mempunyai fungsi yang sama. Dalam kasus ini, KIOSS akan memilih salah satu diantaranya yang memilki daya guna, fungsionalitas, fleksibilitas dan kemudahan penggunaan.

Sebagai tambahan, dimungkinkan juga beberapa komponen digabungkan menjadi satu dalam satu *package* ataupun digabungkan kode-nya sehinga menambah fungsionalitas. Dan ini merupakan hak dari KIOSS. Pada perkembangan berikutnya KIOSS akan mempublikasikan RFC (Request of Comments), untuk memberikan tambahan informasi untuk menyearahkan desain yang lebih sempurna.

Legal License

Tiap autor yang mengirimkan kode harus memasukaan suatu statemen yang menyatakan bahw a kode tersebut merupakan karya original. Dan apabila menggunakan komponen yang membutuhkan lisensi, harus juga menyertakan aturan lisensi yang digunakan.

Kode Sumber & Standarisasi

Author harus memasukkan kode sumber lengkap untuk setiap rutin yang dibuat. Dan seperti halnya dengan kode utama, kode yang digunakan untuk setiap komponen harus mengikuti standar yang diperkenankan di KIOSS.

PENGGUNAAN WHITE SPACE

Baris Kosong

Adanya baris kosong dapat digunakan sebagai pemisah antar sekelompok kode yang berhubungan. Satu baris kosong semestinya juga digunakan pada posisi:
Setelah blok hakcipta, deklarasi *package* dan seksi import.
Antar deklarasi *class*.
Antar deklarasi *methode*.

Spasi Kosong

Object Pascal merupakan suatu pengkodean yang sangat istimewa, bahasa yang mudah dibaca. Pada umumnya, Anda tidak perlu banyak menambahkan spasi kosong dalam membuat kode. Penjelasan dapat digunakan sebagai panduan dalam penggunaan spasi kosong di dalam kode anda.

Spasi kosong yang semestinya tidak digunakan

Antara nama *methode* dan kurung buka.
Sebelum atau sesudah operator . (dot).
Antara operator unary dan operand-nya.
Setelah kurung buka atau sebelum kurung tutup.
Setelah "[" dan sebelum "]".
Sebelum titik koma.

Contoh penggunaan yang benar:

```
function TMyClass.MyFunc(var Value: Integer);
MyPointer               := @MyRecord;
MyClass                 := TMyClass(MyPointer);
MyInteger               := MyIntegerArray[5];
```

Contoh penggunaan yang tidak disarankan:

```
function TMyClass.MyFunc( var Value: Integer ) ;
MyPointer               := @ MyRecord;
MyClass                 := TMyClass ( MyPointer ) ;
MyInteger               := MyIntegerArray [ 5 ] ;
```

Indentasi

Panjang indentasi adalah 2 (dua) spasi untuk semua level indentasi. Dengan kata lain, level pertama indentasi ada dua spasi, indentasi level ke dua adalah empat spasi, level ketiga adalah enam spasi, dan seterusnya. Jangan gunakan karakter Tab.
Namun ada beberapa pengecualian. Beberapa klausa seperti *unit, uses, type, interface, implementation, initialization* dan *finalization* harus rata kiri terhadap margin; atau

diletakkan pada kolom pertama.
Statemen *end* di akhir unit juga harus rata kiri terhadap margin. Pada file proyek, kata *program*, blok *begin* dan *end* utama harus rata kiri pada margin.
Kode di dalam blok *begin...end*, diindentasikan sebanyak dua spasi.

Baris Lanjutan

Maksudnya baris kelanjutan dari statemen yang dituliskan di dalam kode. Satu baris dibatasi maksimal 100 kolom/karakter. Suatu baris perintah yang panjangnya lebih dari 100 kolom seharusnya dipotong untuk kemudian dilanjutkan di baris berikutnya. Jumlah baris maksimal tidak ditentukan, digunakan sesuai kebutuhan saja. Semua baris lanjutan harus diratakan dan diindentasikan terhadap baris pertama, dan indentasi ditentukan sebanya dua spasi. Letakkan pernyataan *begin* pada baris tersendiri.

Contoh:

```
function CreateWindowEx(dwExStyle: DWORD;
  lpClassName: PChar; lpWindowName: PChar;
  dwStyle: DWORD; X, Y, nWidth, nHeight: Integer;
  hWndParent: HWND; hMenu: HMENU; hInstance: HINST;
  lpParam: Pointer): HWND; stdcall;
begin
if ((X = Y) or (Y = X) or
  (Z = P) or (F = J) then
begin
  S := J;
end;
end;
```

Jangan pernah memisahkan antara parameter dengan tipe-nya, kecuali ada pembatas koma diantara parameter. Baris lanjutan jangan dimulai dengan operator binari. Jangan letakkan statemen *begin* di baris yang sama dengan kode.

Contoh:

```
// SALAH
while (Ekspresi1 or Ekspresi2) do begin
  // Perintah
  // PerintahYangLain;
end;

// BENAR
while (Ekspresi1 or Ekspresi2) do
begin
  // Perintah
  // PerintahYangLain;
end;

// SALAH
if (Ekspresi1)
   or (Ekspresi2)
   or (Ekspresi3) then

// BENAR
if (Ekspresi1) or
  (Ekspresi2) or
  (Ekspresi3) then
```


KETERANGAN / KOMENTAR

Bahasa **Object Pascal** mendukung dua jenis komentar: blok dan komentar baris tunggal. Beberapa kegunaan 'petunjuk penggunaan' komentar, diantaranya:
Memudahkan dalam menempatkan keterangan di bagian atas unit untuk menerangkan detil unit tersebut.
Membantu dalam menempatkan keterangan sebelum deklarasi *class*.
Membantu dalam menempatkan keterangan sebelum beberapa deklarasi *methode*.
Ingat bahwa keterangan yang tidak terstruktur lebih buruk daripada tidak diberi keterangan sama sekali.
'Keterangan Sementara' yang akan diubah atau dihilangkan untuk pengkodean berikutnya sebaiknya ditandai dengan Tag khusus "**TODO:**" sehingga akan lebih memudahkan dalam pencarian. Idealnya, keterangan sementara seharusnya telah terhapus semua pada saat aplikasi siap dikirimkan. Keterangan Sementara ini dapat juga difungsikan sebagai penjadwalan pengkodean.

Contoh:

```
// TODO: Ganti dengan fungsi 'Urutkan' jika telah selesai
List.Urut;
```

Keterangan Blok

Object Pascal mendukung dua tipe Keterangan Blok. Keterangan Blok yang sering digunakan adalah pasangan kurung kurawal: { }. Keterangan blok selalu digunakan untuk memberikan keterangan Hak Cipta/ID di awal tiap unit.

Khusus untuk keterangan fungsi dan prosedur memiliki format:

```
//=====================================================================//
procedure TForm1.FormCreate(Sender: TObject);
{ @Deskripsi    : Rutin ini dijalankan ketika form dibuat oleh sistem
  @Parameter    : Sender:TObject
  @Hasil        :
  ----------------------------------------------------------------------}
begin
  // perintah kamu ditulis disini
end;

//=====================================================================//
function TfAplikasi.NamaFungsi( psParameter1:string; piParameter2:integer):word;
{ @Deskripsi    : Fungsi ini digunakan untuk membaca ....
  @Parameter    : psParameter1: nama user
                  piParameter2: kodeAkses
  @Hasil        : 0: sukses
 ----------------------------------------------------------------------}
begin
  // perintah
end;
```

Keterangan Baris Tunggal

Keterangan baris tunggal menggunakan karakter // diikuti oleh teks. Termasuk spasi diantara tanda // dan keterangan yang mengikutinya. Letakkan keterangan di level indentasi yang sama dengan kode yang mengikutinya.

Keterangan baris tunggal juga bisa diletakkan di belakan kode yang dijadikan referensi.
Contoh:

```
if (not IsVisible) then
  Exit;          // nothing to do
Inc(StrLength);  // reserve space for null terminator
```

Keterangan Histori dan Error/Bug

Anda bisa menyisipkan histori pengerjaan dalam baris kode yang anda buat. Hal ini akan memudahkan dalam pelacakan pengembangan.

```
//@H20030207135938 1 prop iDataKu sbg pencarian kode saat ini
    property iDataKu:Integer read FiDataKu write SetiDataKu;
```

Demikian juga dengan BugList

```
//@B20030207140146 0 1 3 Selalu terbaca dgn data owner
procedure TKelasku.SetiDataKu(const Value: Integer);
```

Penyisipan ini dapat dilakukan secara otomatis dengan menggunakan fitur dari aplikasi expert **KIOSS.Assistant**.

Dengan menggunakan aplikasi ini, akan secara otomatis masing-masing penambahan histori dan buglist disisipkan di dalam unit header.

```
  @Histori      : <dibawah ini daftar histori kamu!!!>
    ~H20030207135938 1 prop iDataKu sbg pencarian kode saat ini
  @Bug          : <List dibawah ini daftar error kamu!!!>
    ~B20030207140146 0 1 3 Selalu terbaca dgn data owner
```

Baca seksi Pengorganisasian File diatas.

KELAS (CLASSES)

Pengorganisasian badan Class

Deklarasi badan kelas seharusnya dikelompokkan dalam:
Deklarasi Field
Deklarasi Methode
Deklarasi Property

Field, property dan methods di dalam class seharusnya diurutkan berdasarkan nama.

Access Level

Kecuali kode yang disisipkan oleh Delphi sendiri, ruang lingkup direktif seharusnya dideklarasikan dengan urutan sebagai berikut:
Deklarasi Private
Deklarasi Protected
Deklarasi Public
Deklarasi Published

Secara default level akses adalah published.
Anda seharusnya tidak menggunakan akses public sebagai data. Data seharusnya selalu dideklarasikan di bagian private, dan akses public-nya dilakukan melalui *methods* atau *property*.

Deklarasi *Constructor*

Merupakah suatu kebijakan untuk meletakkan *constructor* dan *destructor* di bagian paling atas dikelompok public.

Deklarasi Methode

Apabila dimungkinkan, deklarasi *method* seharusnya ditampilkan dalam satu baris, perhatikan cara pemenggalan deklarasi *method* pada Bab WhiteSpace.
Contoh:

```
procedure ImageUpdate(Image img, infoflags: Integer,
  x: Integer, y: Integer, w: Integer, h: Integer)
```

ANTAR MUKA (INTERFACE)

Interface dideklarasikan paralel terhadap deklarasi *class*.

```
InterfaceName = interface([Inherited Interface])
InterfaceBody
end;
```

Dan deklarasi *interface* seharusnya diberi indentasi dua spasi. Isi interface diindentasi empat spasi. Dan statement *end* penutup diindentasi dua spasi.

All interface methods are inherently public and abstract; do not explicitly include these keywords in the declaration of an interface method.
Except as otherwise noted, interface declarations follow the same style guidelines as classes.

STATEMENT

Statemen/pernyataan adalah satu baris atau lebih kode yang diikuti dengan titik koma. Statemen yang sederhana memiliki satu titik koma, sedangkan statemen yang memiliki lebih dari satu titik koma merupakan statemen yang komplek

Berikut adalah statemen sederhana:

```
A := B;
```

Dan berikut ini adalah statemen yang komplek:

```
begin
  B := C;
  A := B;
end;
```

Perhatikan bahwa dalam penulisan "**:=**" dilingkungan KIOSS Style sebaiknya diletakkan di kolom ke 37, 47 dan seterusnya. Sehingga akan terlihat:

```
//       1         2         3         4         5
//345678901234567890123456789012345678901234567890123456789012345678901234567890
  bProteksi                          := TIDAK;
  self.ff[ 'uname']                  := uppercase( strMD5( psNamaUser));
  self.ff[ 'passwd']                 := BuatPassword( psNamaUser, psPassword);
  sAkses                             := _BuatAkses( psNamaUser, 0);
  self.ff[ 'hakakses']               := fsAkses;
  self.ff[ 'batas']                  := BuatLimit( psNamaUser, 0);
  fSecurity.DaftarPemakai.Aktif.items[1]      := YA
  fSecurity.DaftarPemakai.AutoDowngrade       := TIDAK
```

Kenapa 37, 47 ....? Yaa... buat kami lebih enak seperti itu.

Statemen Sederhana

Suatu statemen sendiri memiliki titik koma tunggal. Apabila suatu saat diperlukan untuk memenggal statemen, berikan indentasi dua spasi pada baris berikutnya.

```
eMyValue :=
  eMyValue + (SomeVeryLongStatement / OtherLongStatement);

sKodeSalesX := getKodeSales( edt_kodesales.text) +
  edt_kodeSales2.text;
```

Statemen Komplek

Statemen yang komplek selalu diakhiri dengan titik koma, walaupun secara *syntax* tidak diperlukan. Misalnya statemen terakhir pada contoh berikut harus diakhiri dengan titik koma, agar diterima di KIOSS.

```
begin
  MyStatement;
  MyNextStatement;
  MyLastStatement;   // titik koma opsional
end;
```

Statemen Ekspresi/Matematis

Tiap baris seharusnya terdiri hanya satu statemen. Contoh:

```
a := b + c; Inc(Cacah); // SALAH

a := b + c;             // BENAR
Inc(Cacah);             // BENAR
```

Deklarasi Array

Diawali dengan huruf a, dengan huruf kecil

```
aMyArray = array [0..100] of Char;
```

Statemen if

Seharusnya ditampilkan paling tidak dalam dua baris:

```
// TIDAK DIBENARKAN
if iA < iB then LakukanSesuatu;

// BENAR
if iA < iB then
  LakukanSesuatu;
```

Dalam statemen if yang komplek, sebaiknya ditulis seperti ini:

```
// TIDAK DIBENARKAN
if eA < eB then begin
  LakukanSesuatu;
  LakukanSesuatuYangLain;
end else begin
  Begini;
  Beitu;
end;

// BENAR
if eA < eB then
begin
  LakukanSesuatu;
  LakukanSesuatuYangLain;
end
else
begin
  Begini;
  Beitu;
end;
```

Berikut beberapa variasi yang lain:

```
// BENAR
if Kondisi then
begin
  Begini;
end else
begin
  Begitu;
end;

// BENAR
if Condition then
begin
  Begini;
end
else
  Begitu;

// BENAR
if Begini then
begin
  Begini;
end else
  Begitu;
```


Statemen for

```
// TIDAK DIBENARKAN
for i := 0 to 10 do begin
  LakukanSesuatu;
  LakukanSesuatuYangLain;
end;


// BENAR
for i := 0 to 10 do
begin
  LakukanSesuatu;
  LakukanSesuatuYangLain;
end;
```

Statement while

```
// TIDAK DIBENARKAN
while x < j  do begin
  LakukanSesuatu;
  LakukanSesuatuYangLain;
end;


// BENAR
while x < j do
begin
  LakukanSesuatu;
  LakukanSesuatuYangLain;
end;
```

Statemen repeat..until

```
// BENAR
repeat
  x := j;
  j := UpdateValue;
until j > 25;
```

Statemen case

```
// BENAR
case Control.Align of
  alLeft, alNone: NewRange := Max(NewRange, Position);
  alRight: Inc(AlignMargin, Control.Width);
end;


// BENAR
case x of
  csStart:
    begin
      j := UpdateValue;
    end;

  csBegin: x := j;

  csTimeOut:
    begin
      j := x;
      x := UpdateValue;
    end;

end;

// BENAR
```

```
case ScrollCode of
  SB_LINEUP, SB_LINEDOWN:
    begin
      Incr := FIncrement div FLineDiv;
      FinalIncr := FIncrement mod FLineDiv;
      Count := FLineDiv;
    end;
  SB_PAGEUP, SB_PAGEDOWN:
    begin
      Incr := FPageIncrement;
      FinalIncr := Incr mod FPageDiv;
      Incr := Incr div FPageDiv;
      Count := FPageDiv;
    end;
else
  Count := 0;
  Incr := 0;
  FinalIncr := 0;
end;
```

Statemen try

```
// BENAR
try
  try
    EnumThreadWindows(CurrentThreadID, @Disable, 0);
    Result := TaskWindowList;
  except
    EnableTaskWindows(TaskWindowList);
    raise;
  end;
finally
  TaskWindowList := SaveWindowList;
  TaskActiveWindow := SaveActiveWindow;
end;
```


Hungarian Notation for Delphi

Category	Item	Notation Type	Nota-tion	Example
Types & classes	Exception classes	Upper prefix	case E	ESyntax = class (Exception)
	All other types & classes	“	T	TEmployee = class (TObject)

Variabel,kecuali properti (lihat detil #1)	Integer	Lower case prefix	i	iCount: byte;
	Small Integer	“	n	
	Long Integer	“	l	
	Byte	“	by	
	Word	“	w	
	Real (floating point)	“	r	rPrice: Real;
	float	“	f	
	Single	“	f	
	Double	“	d	
	Extended	“	e	
	Comp	“	cmp	
	Char	“	c	cHuruf: Char;
	Pascal string or huge string (2.0)	“	s	sJudul: String[80];
	Null (zero)-terminated string	“	sz	szDescription: array[0..1000] of char;
	Pointer	“	pnt	pntCount: ^iCount;
	pChar	“	pc or lpsz	pcDeskripsi: pChar;
	Boolean	“	b	bSallaried: Boolean;
	Byte Boolean	“	byb	
	Word Boolean	“	wb	
	Long Boolean	“	lb	
	Date/time	“	dt	dtBirthday: TDateTime;
VCL controls	Form	Lower case	f	fKonsumen: TForm;

		prefix		
	Label (see note #2)	"	lbl_	lbl_Status: TDBLabel;
	(other standard controls)	"	(see chart below)	tbl_Go: TSpeedButton;
	(subclassed standard controls)	"	(same as base)	tbl_Go: TMySpeedButton;
	(all other controls, e.g. custom controls for the project at hand)	"	(spell out)	linkedimageUSAMap : TlinkedImage;
	Persistent database field	Delphi-assigned prefix	(table or query name)	tblEmployeesName: TStringField;

Procedures	Procedures, Functions, and Methods (see note #3)	No special notation		procedure ToggleEmployeeStat e (Sender: TObject);

Other identifiers	constants (typed or untyped)	Upper case name with underscores		MAX_HEIGHT: Integer = 24;
	Enumerated type values	Programmer assigned lower case prefix	(2-letter code for item type being enumer-ated)	TVagueness = (vnBefore, vnAfter, vnOn, vnAbout)
	Compiler directive identifiers	Upper case name with underscores		{$DEFINE QA_MODE}

Penamaan Control/Komponen

Penamaan kontrol/komponen dapat dilakukan secara otomatis, jika Anda menggunakan **KIOSS.Assistant**. Dan dapat diedit pada file **kiossPrefixes.txt**
Nama awalan minimal 3 (tiga) huruf, kecuali untuk kondisi tertentu.

Palette Group	Component	Prefix
Standard	TButton	tbl
	TCheckBox	chk
	TComboBox	cbo
	TEdit	edt
	TGroupBox	grp
	TLabel	lbl
	TListBox	lbo
	TMainMenu	mm
	TMemo	mem
	TMenuItem	mnu
	TPanel	pnl
	TPopupMenu	pm
	TRadioButton	rbtn
	TRadioGroup	rgrp
	TScrollBar	sbr
Additional	TBevel	bvl
	TBitBtn	tbl
	TDrawGrid	dgrd
	THeader	hdr
	TImage	img
	TMaskEdit	edt
	TNotebook	nbk
	TOutline	oln
	TScrollBox	sbo
	TShape	shp
	TSpeedButton	sbtn
	TStringGrid	sgrd
	TTabbedNotebook	tbnbk
	TTabSet	tbs
Data Access	TBatchMove	bmv
	TDatabase	db
	TDataSource	ds
	TQuery	qry
	TReport	rpt
	TStoredProc	prc

	TTable	dt
Data Controls	TDBCheckBox	dbchk
	TDBComboBox	dbcbo
	TDBEdit	dbedt
	TDBGrid	dbgrd
	TDBImage	dbimg
	TDBListBox	dblbo
	TDBLookupCombo	dblcbo
	TDBLookupList	dbllbo
	TDBMemo	dbmem
	TDBNavigator	dbnav
	TDBRadioGroup	dbrgrp
	TDBText	dbtxt
Dialogs	TColorDialog	dlgcl
	TFindDialog	dlgfn
	TFontDialog	dlgft
	TOpenDialog	dlgop
	TPrintDialog	dlgpr
	TPrinterSetupDialog	dlgps
	TReplaceDialog	dlgrp
	TSaveDialog	dlgsv
System	TDDEClientConv	ddecc
	TDDEClientItem	ddeci
	TDDEServerConv	ddesc
	TDDEServerItem	ddesi
	TDirectoryListBox	dir
	TDriveComboBox	drv
	TFileListBox	fil
	TFilterComboBox	filt
	TMediaPlayer	mpl
	TOLEContainer	olec
	TPaintBox	pbx
	TTimer	tmr

LIBRARY MAINTENANCE PROCEDURE

Selection Procedure

Update Interval

Version Control

Announcing Update

Bug Reports and Fixes

Patches

Technical Support

Compability

Pustaka

1. Kernighan and Plauger; *The Elements of Programming Style Maguire, Steve; Writing Solid Code*; Microsoft Press, Redmond, WA; 1993, 256 pps.
2. Humphrey, Watts; *A Discipline for Software Engineering*; Addison-Wesley Publishing, New York, NY; 1995, 789 pps. -- A detailed description of the Software Engineering Institute's (SEI) Personal Software Process (PSP).
3. OCDUG Standard
4.