UML软件工程组织

Delphi源程序格式书写规范

tianhaiyise（翻译）

1.规范简介

本规范主要规定Delphi源程序在书写过程中所应遵循的规则及注意事项。编写该规范的目的是使公司软件开发人员的源代码书写习惯保持一致。这样做可以使每一个组员都可以理解其它组员的代码，以便于源代码的二次开发记忆系统的维护。

2.一般格式规范

2.1缩进

缩进就是在当源程序的级改变时为增加可读性而露出的两个空格。缩进的规则为每一级缩进两个空格。不准许使用Tab。因为Tab会因为用户所作的设置不同而产生不同的效果。当遇到begin 或进入判断、循环、异常处理、with语句、记录类型声明、类声明等的时候增加一级， 当遇到end或退出判断、循环、异常处理、with语句、记录类型声明、类声明等的时候减少一级。例如：

```
if TmpInt <> 100 then
   TmpInt := 100;
```

2.2 Begin..End

begin语句和end语句在源程序中要独占一行，例如:

```
for I := 0 to 10 do begin //不正确的用法
end;
for I := 0 to 10 do               //正确的用法
begin
end;
```

2.3空格

在操作符及逻辑判断符号的两端添加空格，例如：I := I + 1;，a and b 等，但添加括号时不需要空格。例如：if (a > b) then //错误的用法

If (a > b) then //正确的用法

又例如：procedure Test(Param1: integer; Param3: string);

3. Object Pascal语法书写格式规范

3.1保留字

Object Pascal 语言的保留字或关键词应全部使用小写字母。

3.2过程和函数

3.2.1命名及格式

过程和函数的名称应全部使用有意义的单词组成，并且所有单词的第一个字母应该使用大写字母。例如：

```
procedure formatharddisk;//不正确的命名
procedure FormatHardDisk;//正确的命名
```

设置变量内容的过程和函数，应使用Set作为前缀，例如：

```
procedure SetUserName;
```

读取变量内容的过程和函数，应使用Get作为前缀，例如：

```
function GetUserName: string;
```

3.2.2 过程和函数的参数

3.2.2.1命名

统一类型的参数写在同一句中:

```
procedure Foo(Param1, Param2, Param3: Integer; Param4: string);
```

3.2.2.2命名

所有参数必须是有意义的；并且当参数名称和其它属性名称重了的时候，加一个前缀‘A’，例如：

```
procedure SomeProc(AUserName: string; AUserAge: integer);
```

3.2.2.3命名冲突

当使用的两个unit中包括一个重名的函数或过程时，那么当你引用这一函数或过程时，将执行在use 子句中后声明的那个unit中的函数或过程。为了避免这种‘uses-clause-dependent’需要在引用函数或过程时，写完整函数或过程的出处。例如：

```
SysUtils.FindClose(SR);
Windows.FindClose(Handle);
```

3.3 变量

3.3.1 变量命名及格式

首先所有变量必须起有意义的名字，使其它组员可以很容易读懂变量所代表的意义，变量命名可以采用同义的英文命名，可使用几个英文单词，但每一单词的首字母必须大写。例如：

```
var
   WriteFormat:: string;
```

同时对于一些特定类型可采用一定的简写如下：

指针类型

```
  P
```

纪录类型

```
  Rec
```

数组类型

```
  Arr
```

类

```
  Class
```

循环控制变量通常使用单一的字符如：i, j, 或 k。 另外使用一个有意义的名字例如：UserIndex ，也是准许的。

3.3.2 局部变量

在过程中使用局部变量遵循所有其它变量的命名规则。

3.3.3 全局变量

尽量不使用全局变量，如必须使用全局变量则必须加前缀‘g’，同时应在变量名称中体现变量的类型。例如：

```
   gprecUserCount: point;//名称为UserCount的全局变量,其类型为指向一结构的指针
```

但是在模块内部可以使用全局变量。所有模块内全局变量必须用‘F’为前缀。如果几个模块之间需要进行资料交换，则需要通过声明属性的方法来实现。例如：

```
type
   TFormOverdraftReturn = class(TForm)
   private
{ Private declarations }
FuserName: string;
FuserCount: Integer;
Procedure SetUserName(Value: string);
Function GetUserName: string;
   public
{ Public declarations }
property UserName: string read GetUserName write SetUserName;
property UserCount: Integer read FuserCount write FuserCount;
   end;
```

3.4类型

3.4.1 大小写协议

保留字的类型名称必须全部小写。Win32 API 的类型通常全部大写，对于其它类型则首字母大写，其余字母小写，例如：

```
var
   MyString: string;     // reserved word
   WindowHandle: HWND; // Win32 API type
   I: Integer;       // type identifier introduced in System unit
```

3.4.2 浮点类型

尽量不使用 Real 类型，他只是为了和旧的Pascal代码兼容，尽量使用Double 类型。Double 类型是对处理器和数据总线做过最优化的并且是IEEE定义的标准数据结构。当数值超出Double的范围时，使用Extended 。但Extended不被Jave支持。但使用其它语言编写的DLL时可能会使用Single 类型。

3.4.3 枚举类型

枚举类型的名字必须有意义并且类型的名字之前要加前缀‘T’。枚举类型的内容的名字必须包含枚举类型名称的简写，例如：

```
TSongType = (stRock, stClassical, stCountry, stAlternative, stHeavyMetal, stRB);
```

3.4.4 数组类型

数组类型的名字必须有意义并且类型的名字之前要加前缀‘T’。如果声明一个指向数组类型的指针必须在该类型的名字之前加前缀‘P’，例如：

```
type
   PCycleArray = ^TCycleArray;
   TCycleArray = array[1..100] of integer;
```

3.4.5记录类型

记录类型的名字必须有意义并且类型的名字之前要加前缀‘T’。如果声明一个指向数组类型的指针必须在该类型的名字之前加前缀‘P’，例如：

```
type
   PEmployee = ^TEmployee;
   TEmployee = record
       EmployeeName: string
       EmployeeRate: Double;
   end;
```

3.5类

3.5.1 命名及格式

类的名字必须有意义并且类型的名字之前要加前缀‘T’。例如：

```
type
   TCustomer = class(TObject)
```

类实例的名字通常是去掉‘T’的类的名字。例如：

```
var
   Customer: TCustomer;
```

3.5.2 类中的变量

3.5.2.1命名及格式

类的名字必须有意义并且类型的名字之前要加前缀‘F’。所有的变量必须是四有的。如果需要从外部访问此变量则需要声明一属性

3.5.3 方法

3.5.3.1命名及格式

同函数和过程的命名及格式。

3.5.3.2 属性访问方法

所有的属性访问方法必须出现在private 或 protected 中。属性访问方法的命名同函数和过程的命名另外读方法(reader method)必须使用前缀‘Get’. 写方法(writer method)必须使用前缀‘Set’。写方法的参数必须命名为‘Value’，其类型同所要写的属性相一致。例如：

```
TSomeClass = class(TObject)
private
   FSomeField: Integer;
protected
   function GetSomeField: Integer;
   procedure SetSomeField( Value: Integer);
public
   property SomeField: Integer read GetSomeField write SetSomeField;
end;
```

3.6属性

3.6.1 命名及格式

同其用操作的，出去前缀‘F’的类的变量的名称相一致 。

3.7文件

3.7.1项目文件

3.7.1.1项目目录结构

程序主目录--Bin（应用程序所在路径）

-Db（本地数据库所在路径）

-Doc（文档所在路径）

-Hlp（帮助文件所在路径）

-Backup（备份路径）

-Tmp（临时文件路径）

3.7.1.2命名

项目文件必须使用一个有意义的名字。例如： Delphi中系统信息的项目文件被命名为 SysInfo.dpr。

3.7.2 Form 文件

3.7.2.1命名

同Form的名称相一致：例如：Form的名称为FormMain则Form文件的名称就为FormMain.frm。

3.7.3 Data Module 文件

3.7.3.1命名

data module文件的命名应该有意义，并且使用‘DM’作为前缀。例如： 用户data module 被命名为‘DMCustomers.dfm’。

3.7.4 Remote Data Module 文件

3.7.4.1 命名

remote data module文件的命名应该有意义，并且使用‘RDM’作为前缀。例如：用户remote data module 被命名为‘RDMCustomers.dfm’。

3.7.5 Unit文件

3.7.5.1普通 Unit

3.7.5.1.1 Unit文件命名

unit文件的命名应该有意义，并且使用‘unit’作为前缀。例如： 通用unit 被命名为‘UnitGeneral’。

3.7.5.2 Form Units

3.7.5.2.1命名

Form unit 文件的名字必须和Form的名称保持一致。例如：主窗体叫FormMain.pas 则Form Unit文件的名字为：UnitFormMain。

3.7.5.3 Data Module Units

3.7.5.3.1命名

Data Module unit 文件的名字必须和Data Module的名称保持一致。例如：主Data Module叫DMMain.pas 则Data Module Unit文件的名字为：UnitDMMain。

3.7.5.4 文件头

在所有文件的头部应写上此文件的用途，作者，日期及输入和输出。例如：

```
{
修改日期：
作者：
用途：
本模块结构组成：
}
```

3.7.6 Forms和Data Modules Forms

3.7.6.1 Form类

1. Form类命名标准

Forms类的命名应该有意义，并且使用‘TForm’作为前缀。例如： About Form类的名字为:

```
TAboutForm = class(TForm)
```

主窗体的名字为

```
TMainForm = class(TForm)
```

2. Form类实例的命名标准

Form 的类实例的名字应同期掉‘T’的Form类的名字相一致。例如：

```
Type Name
  Instance Name

TaboutForm
  AboutForm

TmainForm
  MainForm

TCustomerEntryForm
  CustomerEntryForm
```

3.7.6.2 Data Modules Form

3.7.6.2.1. Data Module Form 命名标准

Data Modules Forms类的命名应该有意义，并且使用‘TDM’作为前缀。例如：

```
TDMCustomer = class(TDataModule)
TDMOrders = class(TDataModule)
```

3.7.6.2.2. Data Module 实例命名标准

Data Module Form 的类实例的名字应同期掉‘T’的Data Module Form类的名字相一致。例如：

Type Name
  Instance Name

TCustomerDataModule
  CustomerDataModule

TordersDataModule
  OrdersDataModule

3.8控件

3.8.1 控件实例的命名

控件的实例应使用去掉‘T’该控件类的名称作为前缀，例如：

输入用户姓名的Tedit的名字为：EditUserName。

3.8.2 控件的简写

控件的名称可使用以下简写，但所用简写于控件名称之间药添加‘_’：

3.8.2.1 Standard Tab

mm TMainMenu
pm TPopupMenu
mmi TMainMenuItem
pmi TPopupMenuItem
lbl TLabel
edt TEdit
mem TMemo
btn TButton
cb TCheckBox
rb TRadioButton
lb TListBox
cb TComboBox
scb TScrollBar
gb TGroupBox
rg TRadioGroup
pnl TPanel
cl TCommandList

3.8.2.2 Additional Tab

bbtn TBitBtn
sb TSpeedButton
me TMaskEdit
sg TStringGrid
dg TDrawGrid
img TImage
shp TShape
bvl TBevel
sbx TScrollBox
clb TCheckListbox
spl TSplitter
stx TStaticText
cht TChart

3.8.2.3 Win32 Tab

tbc TTabControl
pgc TPageControl
il TImageList
re TRichEdit
tbr TTrackBar
prb TProgressBar

ud TUpDown
hk THotKey
ani TAnimate
dtp TDateTimePicker
tv TTreeView
lv TListView
hdr THeaderControl
stb TStatusBar
tlb TToolBar
clb TCoolBar
3.8.2.4 System Tab
tm TTimer
pb TPaintBox
mp TMediaPlayer
olec TOleContainer
ddcc TDDEClientConv
ddci TDDEClientItem
ddsc TDDEServerConv
ddsi TDDEServerItem
3.8.2.5 Internet Tab
csk TClientSocket
ssk TServerSocket
wbd TWebDispatcher
pp TPageProducer
tp TQueryTableProducer
dstp TDataSetTableProducer
nmdt TNMDayTime
nec TNMEcho
nf TNMFinger
nftp TNMFtp
nhttp TNMHttp
nMsg TNMMsg
nmsg TNMMSGServ
nntp TNMNNTP
npop TNMPop3
nuup TNMUUProcessor
smtp TNMSMTP
nst TNMStrm
nsts TNMStrmServ
ntm TNMTime
nudp TNMUdp
psk TPowerSock
ngs TNMGeneralServer
html THtml
url TNMUrl
sml TSimpleMail
3.8.2.6 Data Access Tab
ds TDataSource
tbl TTable
qry TQuery
sp TStoredProc
db TDataBase
ssn TSession
bm TBatchMove
usql TUpdateSQL
3.8.2.7 Data Controls Tab

dbg TDBGrid
dbn TDBNavigator
dbt TDBText
dbe TDBEdit
dbm TDBMemo
dbi TDBImage
dblb TDBListBox
dbcb TDBComboBox
dbch TDBCheckBox
dbrg TDBRadioGroup
dbll TDBLookupListBox
dblc TDBLookupComboBox
dbre TDBRichEdit
dbcg TDBCtrlGrid
dbch TDBChart
3.8.2.8 Decision Cube Tab
dcb TDecisionCube
dcq TDecisionQuery
dcs TDecisionSource
dcp TDecisionPivot
dcg TDecisionGrid
dcgr TDecisionGraph
3.8.2.9 QReport Tab
qr TQuickReport
qrsd TQRSubDetail
qrb TQRBand
qrcb TQRChildBand
qrg TQRGroup
qrl TQRLabel
qrt TQRText
qre TQRExpr
qrs TQRSysData
qrm TQRMemo
qrrt TQRRichText
qrdr TQRDBRichText
qrsh TQRShape
qri TQRImage
qrdi TQRDBMImage
qrcr TQRCompositeReport
qrp TQRPreview
qrch TQRChart
3.8.2.10 Dialogs Tab
OpenDialog TOpenDialog
SaveDialog TSaveDialog
OpenPictureDialog TOpenPictureDialog
SavePictureDialog TSavePictureDialog
FontDialog TFontDialog
ColorDialog TColorDialog
PrintDialog TPrintDialog
PrinterSetupDialog TPrintSetupDialog
FindDialog TFindDialog
ReplaceDialog TReplaceDialog
3.8.2.11 Win31 Tab
dbll TDBLookupList
dblc TDBLookupCombo
ts TTabSet

ol TOutline
tnb TTabbedNoteBook
nb TNoteBook
hdr THeader
flb TFileListBox
dlb TDirectoryListBox
dcb TDriveComboBox
fcb TFilterComboBox
3.8.2.12 Samples Tab
gg TGauge
cg TColorGrid
spb TSpinButton
spe TSpinEdit
dol TDirectoryOutline
cal TCalendar
ibea TIBEventAlerter
3.8.2.13 ActiveX Tab
cfx TChartFX
vsp TVSSpell
flb TF1Book
vtc TVTChart
grp TGraph
3.8.2.14 Midas Tab
prv TProvider
cds TClientDataSet
qcds TQueryClientDataSet
dcom TDCOMConnection
olee TOleEnterpriseConnection
sck TSocketConnection
rms TRemoteServer
mid TmidasConnection

4. 修改规范

本规则所做的规定仅适用于已经纳入配置管理的程序。在这类修改中，要求保留修改前的内容、并标识出修改和新增的内容。并在文件头加入修改人、修改日期、修改说明等必要的信息。

4. 1修改历史记录

对源文件进行经过批准的修改时，修改者应在程序文件头加入修改历史项。在以后的每一次修改时，修改者都必须在该项目中填写下列信息：

修改人
修改时间
修改原因
修改说明即如何修改

4. 2新增代码行

新增代码行的前后应有注释行说明。

// 修改人，修改时间，修改说明
新增代码行
// 修改结束

4. 3删除代码行

删除代码行的前后用注释行说明。

//修改人，修改时间，修改说明
//要删除的代码行（将要删除的语句进行注释）
//修改结束

4. 4修改代码行

修改代码行以删除代码行后在新增代码行的方式进行。

//修改人，修改时间，修改说明
//修改前的代码行
//修改结束

```
        //修改后的代码行
                修改后的代码行
//修改结束
```
