delphi 语法

关键字

absolute 指令（变量）

abstract 指令（方法）

and 运算符（布尔）

array 类型

as 运算符（RTTI）

asm 语句

assembler 向后兼容（汇编）

at 语句（异常处理）

automated 访问类别符（类）

begin 块标记

case 语句

cdecl 函数调用协定

class 类型

const 声明或指令(参数)

constructor 特殊方法

contains 运算符（集合）

default 指令（属性）

destructor 特殊方法

dispid dispinterface 界面类别符

dispinterface 类型

div 运算符

do 语句

downto 语句（for）

dynamic 指令（方法）

else 语句（if 或 case）

end 块标记

except 语句（异常处理）

export 向后兼容（类）

exports 声明

external 指令（函数）

far 向后兼容（类）

file 类型

finalization 单元结构

finally 语句（异常处理）
for 语句
forward 函数指令
function 声明
goto 语句
if 语句
implementation 单元结构
implements 指令（属性）
in 运算符（集合）- 工程结构
index 指令（dipinterface 界面）
inherited 语句
initialization 单元结构
inline 向后兼容（见 asm）
interface 类型
is 运算符（RTTI）
label 声明
library 程序结构
message 指令（方法）
mod 运算符（数学）
name 指令（函数）

near 向后兼容（类）
nil 数值
nodefault 指令（属性）
not 运算符（布尔）
object 向后兼容（类）
of 语句（case）
on 语句（异常处理）
or 运算符（布尔）
out 指令（参数）
overload 函数指令
override 函数指令
package 程序结构（控件包）
packed 指令（记录）
pascal 函数调用协定
private 访问类别符（class）
procedure 声明

program 程序结构
property 声明
protected 访问类别符（类）
public 访问类别符（类）
published 访问类别符（类）
raise 语句（异常处理）
read 属性类别符
readonly dispatch 界面类别符
record 类型
register 函数调用协定
reintroduce 函数指令
repeat 语句
requires 程序结构（控件包）
resident 指令（函数）
resourcestring 类型
safecall 函数调用协定
set 类型
shl 运算符（数学）
shr 运算符（数学）
stdcall 函数调用协定

stored 指令（属性）
string 类型
then 语句（if）
threadvar 声明
to 语句（for）
try 语句（异常处理）
type 声明
unit 单元结构
until 语句
uses 单元结构
var 声明
virtual 指令（方法）
while 语句
with 语句
write 属性类别符
writeonly dispatch 界面类别符

xor 运算符（布尔）

优先法则

单目运算符（最高优先级）

@ 取变量或函数的地址(返回一个指针)

not 逻辑取反或按位取反

乘除及按位运算符

* 相乘或集合交集

/ 浮点相除

div 整数相除

mod 取模（整数相除的余数）

as 程序运行阶段类型转换（RTTI 运算符）

and 逻辑或按位求和

shl 按位左移

shr 按位右移

加减运算符

+ 相加、集合并集、字符串连接或指针增加一个偏移量

- 相减、集合差集或指针减少一个偏移量

or 逻辑或按位或运算

xor 逻辑或按位异或运算

关系及比较运算符(最低优先级)

= 判断是否相等

<> 判断是否不相等

< 判断是否小于

> 判断是否大于

<= 判断是否小于或等于,或是否是一个集合的子集

>= 判断是否大于或等于,或是否是一个集合的父集

in 判断是否是集合成员

is 判断对象是否类型兼容（又一个 RTTI 运算符）

Boolean (ByteBool WordBool LongBool)

Char

Integer

ShortInt -128~127

Byte 0~255

SmallInt -32768~32767

Word 0~65535

LongInt LongWord

Int64

Interger Cardinal

AnsiChar

WidChar

#=Chr(),把数字转化为字符，Ord 为相反，把字符转化为数字

#9 Tab 键

#10 换行

#13 回车

类型转换

Chr 将一个有序数据转换为一个 ANSI 字符

Ord 将一个有序类型值转换为它的序号

Round 转换一个实型值为四舍五入后的整型值

Trunc 转换一个实型值为小数截断后的整型值

Int 返回浮点数的整数部分

IntToStr 将数值转换为字符串

IntToHex 将数值转换为十六进制数字符串

StrToInt 将字符串转换为一个整型数，如字符串不是一个合法的整型将引发异常

StrToIntDef 将字符串转换为一个整数，如字符串不合法返回一个缺省值

Val 将字符串转换为一个数字（传统 Turbo Pascal 例程用于向后兼容）

Str 将数字转换为格式化字符串（传统 Turbo Pascal 例程用于向后兼容）

StrPas 将零终止字符串转换为 Pascal 类型字符串，在 32 位 Delphi 中这种类型转换是自动进行的

StrPCopy 拷贝一个 Pascal 类型字符串到一个零终止字符串，在 32 位 Delphi 中这种类型转换是自动进行的

StrPLCopy 拷贝 Pascal 类型字符串的一部分到一个零终止字符串

FloatToDecimal 将一个浮点数转换为包含指数、数字及符号的十进制浮点记录类型

FloatToStr 将浮点值转换为缺省格式的字符串

FloatToStrF 将浮点值转换为特定格式的字符串

FloatToText 使用特定格式，将一个浮点值拷贝到一个字符串缓冲区

FloatToTextFmt 同上面例程，使用特定格式，将一个浮点值拷贝到一个字符串缓冲区

StrToFloat 将一个 Pascal 字符串转换为浮点数

TextToFloat 将一个零终止字符串转换为浮点数

Dec 将例程中的参数值递减 1 或一个特定的值，其中特定值可在第二个可选参数中定义

Inc 将例程中的参数值增加 1 或一个特定的值

Odd 如果参数为奇数返回真

Pred 根据参数在其数据类型定义中的序列，返回参数值的前驱值

Succ 返回参数值的后继值

Ord 返回参数值在其数据类型值集合中的序号

Low 返回参数对应的有序数据类型的最小取值

High 返回参数对应的有序数据类型的最大取值

实数：

Single

Double

Extended

Real {$REALCOMPATIBILITY ON}据说不标准，要兼容旧的格式使用这个宏

Comp

Currency

TDateTime = type Double

为了后续使用或直接用于变量，需要给自定义类型命名。如果自定义一个命名的类型，你必须将代码放在特定的 type 区，如下所示：

最多的如定义类等。

```
type
Uppercase = 'A'..'Z';
Temperatures = array [1..24] of Integer;
Date = record
Month: Byte;
Day: Byte;
Year: Integer;
end;
Colors = (Red, Yellow, Green, Cyan, Blue, Violet);
Letters = set of Char;
end;
```

子界类型

a...b

a~b 为有序类型

枚举类型

```
type
Colors = (Red, Yellow, Green);//园括弧括起来的，在属性列表中最多出现。
end;
```

只能有一个值

CPP = enum

集合类型

```
type Letters = set of Char;
var Letters1,Letters2etters;
begin
Letters1 := ['A'.'B'.'C'];
Letters2 := ['K'];
end;
```

是什么什么的集合，可以有多个属性。

CPP = 多字段与或


```
Font.style := [fsBold, fsItalic]; // two styles
Font.style :=Oldstyle + [fsBold, fsItalic];
```

数组类型

```
type
MonthTemps = array [1..24, 1..31] of Integer;
YearTemps = array [Jan..Dec] of Integer;
```

记录类型

```
type
Date = record
Year: Integer;
Month: Byte;
Day: Byte;
end;
```

类似 structure

指针

```
type
PointerToInt = ^Integer;
var
P:^Integerl
X:Integer;
begin
P := @X; p = &x;
X :=10; x = 10;
P^ :=20; *p = 20;
end;
```

除了表示已分配内存的地址外，指针还能通过 New 例程在堆中动态分配内存，不过当你不需要这个指针时，你也必须调用 Dispose 例程释放你动态分配的内存。

```
P:=nil;
```

Pointer 类型，如 void *;

实际上，Delphi 中必须使用指针的情况很少，这是 Delphi 开发环境一个诱人的优点。

文件类型

```
type
IntFile = file of Integer;
```

能打开一个与这个结构相应的物理文件、向文件中写入整数、或者从文件中读取当前的值

简单语句：

单行：X:= Y+Z;

调用一个过程

```
Randomize;
```

复合语句 begin end;

赋值 :=

=表示关系运算符，用于判断是否相等用 c: ==

条件

if 用and、 or 、 not等布尔操作符联接起来的条件 then
statement;
else
statement;
end;

Case语句
case Number of
1: Text:='One';
2: Text:='Two';
else //default;
Text:='Unknow';
end;

循环 循环条件为顺序数
for I:=1 to 10 do
statement;

for I := 10 downto 1 do 递减
statement;

while conditions do
statement;

repeat
statement;
until conditions;

With语句
begin
with BirthDay do
begin
Year:=1999;
Month :=2;
Day :=14;

```
end;
end;
```

减少 BirthDay.xxx 的代码工作量，但现在的自动完成比他效率高

可以 with 多个，同样属性的取最后一个的，容易出错。

函数和过程

```
procedure Hello;
begin
ShowMessage ('Hello world!');
end;
```

调用：Hello;

```
function Double (value: Integer) : Integer;
begin
Double := value * 2; //函数名做返回值
end;

function Double2 (value: Integer) : Integer;
begin
Result := value * 2; //Result 作返回值
end;
```


引用参数

同 C++中的(&xx)

使用 var 关键字表示

```
procedure Produ(var value:Integer);
```

既做传递参数，又把值返回给调用程序。

```
procedure Produ(out value:Interger);
```

用作返回。

常量参数

使用 const

开放数组参数

```
function Sum (const A: array of Integer): Integer;
var
I: Integer;
begin
Result := 0;
for I := Low(A) to High(A) do
Result := Result + A[I];
end;
```

可以传递大小不定的数组

类型变化的开放数组参数

可以接收多种类型的数组参数

```
function SumAll (const Args: array of const): Extended;
```

调用协定

fastcall 只要有可能，传递到CPU 寄存器多达 3 个，操作更快

delphi3+ 缺省标记

win32 API 是 stdcall,win16 API 是原始的 Pascal 调用和 cdecl 调用混合体

方法：

是一种特殊的函数或过程

他与类这一数据类型相对应，每个事件都定义一个方法，通常是过程。

forward 申明

欲声明一个过程或函数，而且只给出它的名字和参数，不列出其实现代码，需要在句尾加 forward 关键字：

```
procedure Hello; forward;
```

可用来写递归调用

当你在一个单元（关于单元的更多内容见下一章）的 interface 部分声明一个过程或一个函数时，它被认为是一个 forward 声明，即使没有 forward 关键字也一样。

过程类型

```
type
```

```
IntProc = procedure (var Num: Integer);

IP: IntProc;
begin
IP := otherProcedure;
```

函数重载 overload关键字

```
function Min (A,B: Integer): Integer; overload;
function Min (A,B: Int64): Int64; overload;
function Min (A,B: Single): Single; overload;
function Min (A,B: Double): Double; overload;
function Min (A,B: Extended): Extended; overload;
```

确省参数

```
procedure MessBox (Msg: string;
Caption: string = 'Warning';
Flags: LongInt = mb_OK or mb_IconHand);
```


字符串

ShortString

AnsiString

WideSting

如果只简单地用String定义字符串,那么该字符串可能是短字符串也可能是ANSI长字符串,这取决于$H 编译指令的值，$H+（确省）代表长字符串（ANSIString 类型）。长字符串是Delphi 库中控件使用的字符串。

SetLength (String1, 200);设置长度分配内存

类型转换,PChar(String)

```
SetWindowText (Handle, PChar (Label1.Caption));
GetWindowText (Handle, PChar (S1), Length (S1));
```

格式化字符串

```
format ('First %d, Second %d', [n1, n2]);//集合
```

d(decimal) 将整型值转换为十进制数字字符串

x (hexadecimal) 将整型值转换为十六进制数字字符串

p (pointer) 将指针值转换为十六进制数字字符串
s (string) 拷贝字符串、字符、或字符指针值到一个输出字符串
e (exponential) 将浮点值转换为指数表示的字符串
f (floating point) 将浮点值转换为浮点表示的字符串
g (general) 使用浮点或指数将浮点值转换为最短的十进制字符串
n (number) 将浮点值转换为带千位分隔符的浮点值
m (money) 将浮点值转换为现金数量表示的字符串，转换结果取决于地域设置,详见 Delphi 帮助文件的 Currency and date/time formatting variables 主题

内存
动态数组
```
Array1: array of Integer;
SetLength(Array1, 100);
```
下标从 0 开始

普通数组下标可以随便写
动态数组不行，Length,Hight,Low 函数了解动态数组状况
```
for I := Low (Array1) to High (Array1) do
Array1 [I] := I;
```


windows 相关
```
type
THandle = LongWord;
```

外部声明
```
// forward declaration
function LineTo (DC: HDC; X, Y: Integer): BOOL; stdcall;
// external declaration (instead of actual code)取代真实代码
function LineTo; external 'gdi32.dll' name 'LineTo';
```
这段声明表示函数 LineTo 的代码同名保存在 GDI32.DLL 动态链接库中（最重要的 Windows 系统库之一）。实际应用时，外部声明中的函数名与 DLL 中的函数名可以不同。
与 WindowsAPI 对应的声明：
```
BOOL EnumWindows(
WNDENUMPROC lpEnumFunc,LPARAM lParam
);
```
＝》

```
function EnumWindows (lpEnumFunc: TFNWndEnumProc,lParam: LPARAM): BOOL; stdcall;
```

回调函数

```
BOOL CALLBACK EnumWindowsProc (
HWND hwnd, // handle of parent window
LPARAM lParam // application-defined value
);
=>
type
EnumWindowsProc = function (Hwnd: THandle, Param: Pointer): Boolean; stdcall;
```

直接使用方法

Variant 变量没有类型

运算效率低

单元 Unit

```
unit unitName;
interface
// other units we need to refer to
uses
A, B, C;
// exported type definition
type
newType = TypeDefinition;
// exported constants
const
Zero = 0;
// global variables
var
Total: Integer;
// list of exported functions and procedures
procedure MyProc;
implementation
uses
```

```
D, E;
// hidden global variable
var
PartialTotal: Integer;
// all the exported functions must be coded
procedure MyProc;
begin
// ... code of procedure MyProc
end;
initialization
// optional initialization part
finalization
// optional clean-up code
end.
```

单元用作命名空间

Delphi 应用程序源代码文件可分成两类，它们是一个或多个单元文件及一个程序文件，单元文件可以看成是次一级的文件，它被应用程序的主体——程序文件——引用。

```
program Project1;
```



```
uses
forms,
Unit1 in Unit1.PAS?{form1Dateform};

begin
Application.Initialize;
Application.Createform (Tform1, form1);
Application.Run;
end.
```

面向对象

```
type
TData = class
A:Integer;
```

```
B:Char;
function F(V:Integer):Boolean;
var C:Integer;
begin
A :=1;
C :=A;
end;
procedure P(V:Char);
begin
B:='b';
end;
end.
```

public
private
protected

self 变量
publish

继承
type
MyClass = class(BaseClass);
多继承
type
MyClass = class();

构造器
constructor 代替 procedure;自动分配内存
对象方法重载 同函数

多态
type
BaseClass = class
public
function F:String;virtual;

```
end;

type
MyClass = class(BaseClass)
public
function F:String;override;
end;
```

类型兼容性
向上兼容
之类兼容父类

类型信息
`if MyClass is BaseClass then`

继承
`inherited;` 使用父辈被继承的方法

类引用
`MyClassClass = class of MyClass;`

`Create()`
`Free()`
type 里面类声明
implementation
实现

```
initialization //初始化段
Init();
finalization //反初始化段
UnInit();

constructor Create();
destructor Destroy();
```

```
procedure fly(); virtual; abstract;抽象
```

接口 interface

```
type
Icanfly = interface
['{10000000-0000-0000-0000-0000000000}']
function Fly:String;
end;

type
TAirplane = class(TomerfaceObject,IcanFly)
function Fly:String virtual;
end;

Flyer1 :=TAirplance.Create as Icanfly;

QueryInterface
```


异常 错误处理

```
raise Exceptoion.Create('a error');

try
A();
except
...
finally
...
end;
```

继承异常类 p54

VCL

TObject

TPresistent

TComponent

TControl 可视化控件

Windowsed Controls 基于 window 的控件(TWinControl)

Nonwindowsed Controls(TGraphicsControl)

Nonvisual Components 非可视控件

可视化组件

Window-based controls

有 GDI 句柄，Windows 管理，拥有 Windows 对象的属性

Graphical controls

无 GDI 句柄，自行绘图管理，如大量的 Laber(比较 Static 控件) speedbutton，节省 GDI 资源,但响应事件麻烦。

novisiual compoenets 一些管理组件

可融入编辑环境

published private/public

procedure p:string;

property month:Integer

read FMonth write SetMonth;

读写为变量或者过程。

read GetMonth write SetMonth;

function GetMonth:Integer;读

procedure SetMonth(const value:Integer);写

只读属性的话就不带 read 或 write

对象拷贝

Assigned()

事件

```
private
FonClick : TNotifyEvent;
.....
published:
property onClick : TNotifyEvent Read FonClick Write FonClick;
```