แผนปฏิบัติการสอนสัปดาห์ที่ 3

วิชา 030143361 การโปรแกรมคอมพิวเตอร์สำหรับงานควบคุม **ระดับ:** ปริญญาตรี

เรื่อง Function และ Procedure **เวลา:** บรรยาย 120 นาที
ปฏิบัติ 120 นาที

ก. วัตถุประสงค์การสอน	**รายละเอียดตามที่ระบุไว้ใน**
1. เขียน Function ที่มีและไม่มีการส่ง Parameters และการคืนค่ากลับ	หน้า 34-46
2. เขียน Procedure ที่มีและไม่มีการส่ง Parameters	
3. Procedure ที่ตอบสนองจาก Events	
4. เรียกใช้งาน Procedures และ functions	
5. ดักจับ Error ด้วย try except และ try finally	

ข. การจัดการเรียนการสอน

เวลา – นาที		0	60	120	180	240
วัตถุประสงค์		1, 2, 3, 4, 5				
การนำเข้าสู่บทเรียน						
ให้เนื้อหา						
สรุปเนื้อหา						
พัก						
ทำแบบฝึกหัด						
ให้เนื้อหา						
สรุปเนื้อหา						
ทดสอบและเก็บคะแนน						
ประเมินผล		พิจารณาจากผลการทดสอบในชั่วโมงสุดท้าย				
วิธีการสอน:	บรรยาย					
	ถาม – ตอบ					
	ทำแบบฝึกหัด					
	บรรยาย และแสดงให้ดู					
	ทดสอบ					
สื่อการสอน:	คอมพิวเตอร์					
	Presentations					

Content

1. เขียน Function ที่มีและไม่มีการส่ง Parameters และการคืนค่ากลับ

ในภาษา Pascal มีการแยกเขียน Function และ Procedure ไว้อย่างชัดเจน Function หมายถึง ชุดคำสั่งที่เมื่อถูกเรียกใช้แล้วจะมีข้อมูลส่งคืนกลับไปให้ผู้เรียกใช้ เช่นต้องการบวกเลข 2 ชุด คือ 1 กับ 3 เมื่อเรียกใช้ Function และส่งค่า Parameter 1 กับ 3 เข้าไปบวกกันใน Function แล้วคืนคำตอบกลับมาคือ 4 รูปแบบของ Function อาจมีหรือไม่มีการรับค่า Parameter ก็ได้ หรือถ้ามีการรับค่า Parameter อาจรับได้ตั้งแต่ 1 ตัวไปจนถึงหลายตัว หรือรับมาเป็น Object เลยก็ได้ แต่ต้องมีการคืนค่ากลับ 1 ค่าเสมอ ค่าที่คืนกลับอาจเป็นข้อมูลเดี่ยวๆ หรือ Array หรือ Object ก็ได้

การประกาศ Function

```
function ชื่อ(ข้อมูลเข้า:ชนิดข้อมูลเข้า):ชนิดข้อมูลออก;
```

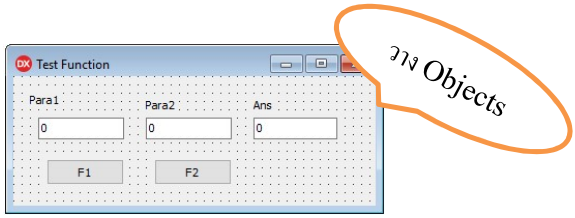
Example

```
function AddNum(a, b:Integer):Integer;  // รับค่า a, b เป็นจำนวนเต็ม และคืนค่าเป็นจำนวนเต็ม
function GetPara1():String;     // ไม่รับค่า และคืนค่าเป็นตัวหนังสือ
function Cal1(a:Integer; b:Single):String; // รับค่า a เป็นจำนวนเต็ม b เป็นจำนวนทศนิยม และคืนค่าเป็นตัวหนังสือ
```

Tip

- การประกาศตัวแปรมีรูปแบบดังนี้ ชื่อตัวแปร:ชนิดตัวแปร;
- เมื่อประกาศ Function ที่รับพารามิเตอร์หลายตัวแต่ชนิดเดียวกันให้คั่นด้วย ,
- ถ้าพารามิเตอร์หลายตัวคนชนิดเดียวกันให้ทำการประกาศใหม่และให้คั่นด้วย ;

Example




```
unit UFunc1;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils,
System.Variants, System.Classes, Vcl.Graphics,
 Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

type
  TFrm1 = class(TForm)
    BtF1: TButton;
    BtF2: TButton;
    EdPara1: TEdit;
```

ตั้งชื่อ Object

```
  EdPara2: TEdit;
  EdAns: TEdit;
  Label1: TLabel;
  Label2: TLabel;
  Label3: TLabel;
  procedure BtF1Click(Sender: TObject);
  procedure BtF2Click(Sender: TObject);
 private
  function AddNum(a, b:Integer):Integer;
  function GetPara1():String;
   { Private declarations }
 public
   { Public declarations }
 end;
var
 Frm1: TFrm1;
implementation

{$R *.dfm}

function TFrm1.AddNum(a, b: Integer): Integer;
begin
 Result:=a+b;
end;
```

ประกาศ Function ใต้ private

รับค่า Parameters

```
procedure TFrm1.BtF1Click(Sender: TObject);
var
 In1, In2, Ans:Integer;
begin
 In1:=StrToInt(EdPara1.T
 In2:=StrToInt(EdPara2.Text);
 Ans:=AddNum(In1, In2);
 EdAns.Text:=IntToStr(Ans);
end;

procedure TFrm1.BtF2Click(Sender: TObject);
begin
 EdAns.Text:=GetPara1();
end;

function TFrm1.GetPara1: String;
begin
 Result:=EdPara1.Text;
end;

end.
```

ประกาศตัวแปรใต้ var

ส่งค่า Parameters

รับคำตอบกลับ

Code\week3\ Ex_Func_1

ให้เพิ่ม Function SubNum(a, b: Integer): Integer; แล้วเพิ่มปุ่มอีกปุ่มสำหรับคำนวณค่าลบกัน

Tip

- เมื่อประกาศ Function ใต้ private แล้วให้กด ctrl+c โปรแกรมจะสร้าง Body ให้โดยอัตโนมัติ
- การประกาศตัวแปรภายใน Local ให้ประกาศ var ก่อน
- การแปลงชนิดข้อมูลจาก String เป็น Integer ใช้ Function StrToInt()
- การแปลงชนิดข้อมูลจาก Integer เป็น String ใช้ Function IntToStr()
- การเรียกใช้งาน Function คือ ตัวแปร:=ชื่อFunction(Para1, Para2);

- ถ้าไม่ต้องการค่าที่คืนกลับสามารถเรียกใช้Functionโดยตรงได้ เช่น ชื่อFunction(Para1, Para2);

2. เขียน Procedure ที่มีและไม่มีการส่ง Parameters

การเขียน Procedure มีการรับหรือไม่รับค่า Parameters เช่นเดียวกับ Function แต่จะไม่มีการคืนค่าใดๆกลับ Procedure จะทำงานเสร็จสิ้นที่ตัวเองเลย

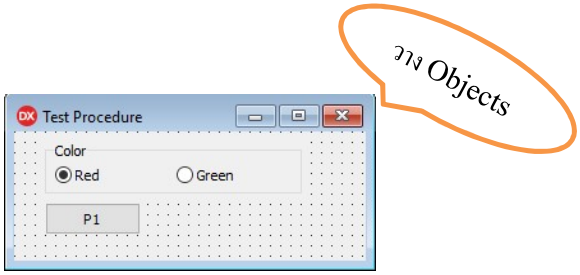
การประกาศ Procedure

```
proceudre ชื่อ(ข้อมูลเข้า:ชนิดข้อมูลเข้า);
```

Example

```
procedure FrmYellow();        // ไม่รับค่า และไม่คืนค่าใดๆ
procedure FrmColor(Cl:TColor);        // รับค่าเป็นตัวแปรสี
procedure FrmColor2(a:Integer; Cl:TColor);        // รับค่าเป็นตัวแปรจำนวนนับ และสี
```

Example




```
unit UProc1;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils,
System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls,
Vcl.ExtCtrls;

type
  TFrm1 = class(TForm)
    BtP1: TButton;
    Rg1: TRadioGroup;
    procedure BtP1Click(Sender: TObject);
    procedure Rg1Click(Sender: TObject);
  private
    procedure FrmYellow();
    procedure FrmColor(Cl:TColor);
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Frm1: TFrm1;

implementation

{$R *.dfm}
```

ประกาศ Procedure

```
{ TFrm1 }

procedure TFrm1.BtP1Click(Sender: TObject);
begin
 FrmYellow;
end;

procedure TFrm1.FrmColor(Cl: TColor);
begin
 Frm1.Color:=Cl;
end;

procedure TFrm1.FrmYellow;
begin
 Frm1.Color:=ClYellow;
end;

procedure TFrm1.Rg1Click(Sender: TObject);
begin
 if Rg1.ItemIndex = 0 then
  FrmColor(ClRed)
 else if Rg1.ItemIndex = 1 then
  FrmColor(ClGreen);
end;

end.
```

รับค่าสีมาใช้

ไม่รับค่าใดๆ

Code\week3\ Ex_Proc_1

ในกรณีที่ต้องการให้ Procedure หรือ Function ที่มีชื่อเดียวกันแต่ทำงานไม่เหมือนกันขึ้นอยู่กับค่า Parameter ที่ส่งมา สามารถใช้การ Overload ได้ เพื่อให้หลาย Function ที่มีชื่อเดียวกันทำงานร่วมกันได้

Example

```
function SumAsStr(a, b :integer): string; overload;
begin
   Result := IntToStr(a + b) ;
end;
function SumAsStr(a, b : extended; Digits:integer): string; overload;
begin
   Result := FloatToStrF(a + b, ffFixed, 18, Digits) ;
end;
```

ใส่ overload ไว้ท้ายของทั้ง 2 Function

ถ้า Function ที่มีชื่อเหมือนกันอยู่คนละ Unit ที่ไม่ได้ทำ Overload การเรียกใช้งานจะต้องอ้างชื่อ Unit นั้นก่อน เช่น

Example

```
unit B;
...
uses A;
...
```

```
procedure RoutineName;
begin
  Result := A.RoutineName;
end;
```

Parameter ของ Function และ Procedure สามารถตั้งค่า Default ไว้ล่วงหน้าได้ตามตัวอย่างด้านล่าง การเรียกใช้นั้นสามารถเลือกใส่หรือไม่ใส่ค่า Parameter นั้นก็ได้ ถ้าไม่ได้ใส่ค่าโปรแกรมจะนำค่า Default มาใช้งาน

Example

```
//การประกาศตัวแปรใน Function ที่การใส่ค่า Default
function SumAsStr (a,b : extended; Digits : integer = 2) : string;

//การเรียกใช้
 SumAsStr(6.0, 3.0);
 SumAsStr(6.0, 3.0, 2);

//สามารถทำ Overload เฉพาะค่า Default ได้
 procedure DoIt(A:extended; B:integer = 0) ; overload;
procedure DoIt(A:extended) ; overload;
```

https://www.thoughtco.com/understanding-method-overloading-and-default-parameters-1058217

การส่งค่า Parameter ให้กับ Function หรือ Procedure นั้นสามารถส่งได้ 4 แบบดังนี้

Key	Details
	ส่งค่า Copy ของต้นฉบับไป ถ้ามีการแก้ไขตัวแปรจะไม่ส่งผลต่อต้นฉบับ
var	ส่งค่าต้นฉบับไป ถ้ามีการแก้ไขตัวแปรจะเปลี่ยนค่าต้นฉบับด้วย
const	ส่งเป็นค่าคงที่ไป (Read Only) ถ้ามีการแก้ไขตัวแปรจะไม่ส่งผลต่อต้นฉบับ
out	ส่งค่าต้นฉบับไปเหมือน var แต่จะส่งค่าตัวแปรที่ยังไม่ได้ให้ค่าเริ่มต้นไป ส่วนใหญ่จะใช้ใน distributed-object models เช่น COM

Example

```
(X, Y: Real)
(var S: string; X: Integer)
(HWnd: Integer; Text, Caption: PChar; Flags: Integer)
(const P; I: Integer)
(out Info: SomeRecordType)
```

เมื่อประกาศตัวแปร var, const หรือ const สามารถยกเลิกการประกาศชนิดข้อมูลได้ เช่น

Example

```
procedure TakeAnything(const C);
```

ก่อนนำตัวแปรที่ไม่ได้ประกาศชนิดข้อมูลไปใช้จำเป็นต้องแปลงเป็นชนิดข้อมูลที่ต้องการด้วยวิธีการ casting ก่อน

Example

```
function Equal(var Source, Dest; Size: Integer): Boolean;
type
  TBytes=array[0..MaxInt - 1] of Byte;
var
  N:Integer;
begin
  N:=0;
  while (N < Size) and (TBytes(Dest)[N] = TBytes(Source)[N]) do
    Inc(N);
  Equal:=N = Size;
end;
```

Tip

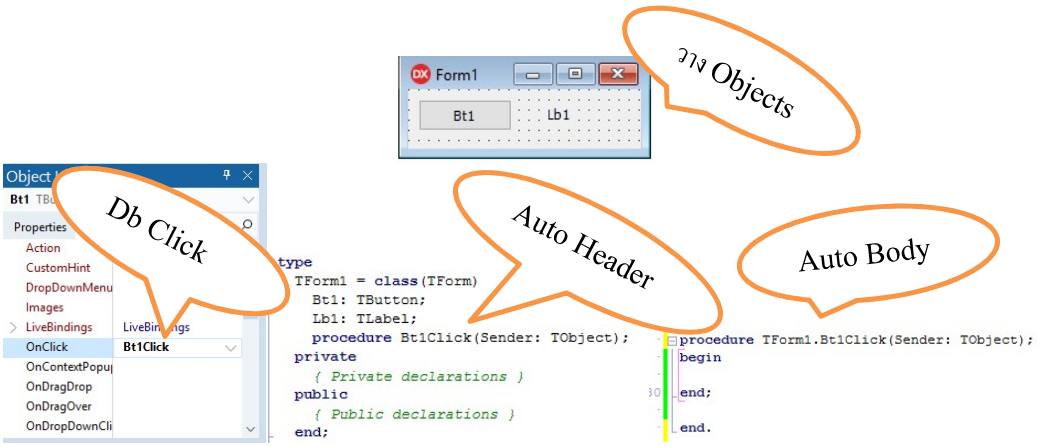
- ยังมีการส่งค่า Parameter แบบ Array, Variant และอื่นๆอีก สามารถอ่านเพิ่มเติมได้จาก Link ด้านล่าง

http://docwiki.embarcadero.com/RADStudio/Rio/en/Parameters_(Delphi)

3. Procedure ที่ตอบสนองจาก Events

เมื่อมีการกำหนด Event ให้กับ Object แล้วโดยการ Double Click ในช่องว่างหลังชื่อ Event โปรแกรมจะสร้าง Header และ Body ให้เองโดยอัตโนมัติ การทำงานของ Procedure ประเภทนี้จะถูกเรียกใช้โดยเหตุการณ์ต่างๆที่เกิดขึ้นกับ Object นั้นๆเช่น OnClick หมายถึงเมื่อมีการ Click Left Mouse ลงไปบนตัว Object โปรแกรมจะโดดเข้ามาทำงานคำสั่งที่อยู่ภายใน Event ที่สร้างขึ้นนี้ทันที โดยที่จะมีการส่งค่า Parameter มาเป็น Object ที่ถูกกระทำมาให้เป็นข้อมูลชนิด TObject ชื่อว่า Sender ซึ่งเป็น Class บรรพบุรุษของ Object ทุกตัวทำให้สามารถแปลงเป็น Object ใดๆก็ได้ ในบาง Event ยังมีการส่งค่า Parameter อื่นๆ มาด้วย เช่น OnMouseDown จะมีการส่งค่าพิกัดที่คลิก Mouse ลงไปมาด้วยเพื่อให้ Programmer สามารถนำไปใช้ได้

Example




```
unit UProcEvt1;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils,
System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

type
  TForm1 = class(TForm)
    Bt1: TButton;
    Lb1: TLabel;
    procedure Bt1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

สวมรอยเปลี่ยน Property

```
var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Bt1Click(Sender: TObject);
var
 x:TLabel;
begin
 if Sender is TButton then
  TButton(Sender).Caption:='Click';

 x:=FindComponent('Lb1') as TLabel;     // Find By Name
 if Assigned(x) then
  x.Caption:='123';
end;

end.
```

เช็คชนิด Object

หา Object

สวมรอยเปลี่ยน Property

Code\week3\ Ex_Proc_Event_1

4. เรียกใช้งาน Procedures และ functions

ก่อนสร้าง Procedure หรือ Function ต้องคิดก่อนว่าการทำงานที่เราสร้างนั้นต้องการค่าที่คืนกลับไหม เช่น ถามว่า 2+3 เท่ากับเท่าไหร่ อันนี้แน่นอนว่าเราต้องการคำตอบว่า 5 ดังนั้นจึงต้องสร้างเป็น Function แต่ถ้าเป็นการสั่งให้ไปทำงานใดๆแล้วจบ เช่น “ปิดไฟ” ถ้าเราไม่ต้องการคำตอบว่าปิดได้ไหมก็ให้สร้างเป็น Procedure แต่ถ้าเราต้องการคำตอบกลับมาด้วยว่าปิดได้ไหม ให้สร้างเป็น Function แล้วคืนค่า Boolean กลับมา การเรียกใช้งานนั้นชัดเจนอยู่แล้วว่า Function จะต้องมีตัวแปรมารับค่าที่คืนกลับ หรือถ้าไม่ต้องการใช้ค่าที่คืนกลับก็สามารถเรียกใช้ตรงๆเหมือน Procedure ได้เช่นกัน ในกรณีที่ Function หรือ Procedure ต้องการรับ Parameter การเรียกใช้จำเป็นต้องส่งค่า Parameter ดังกล่าวไปให้ครบด้วย ยกเว้น Function นั้นมีการตั้งค่า Default ไว้

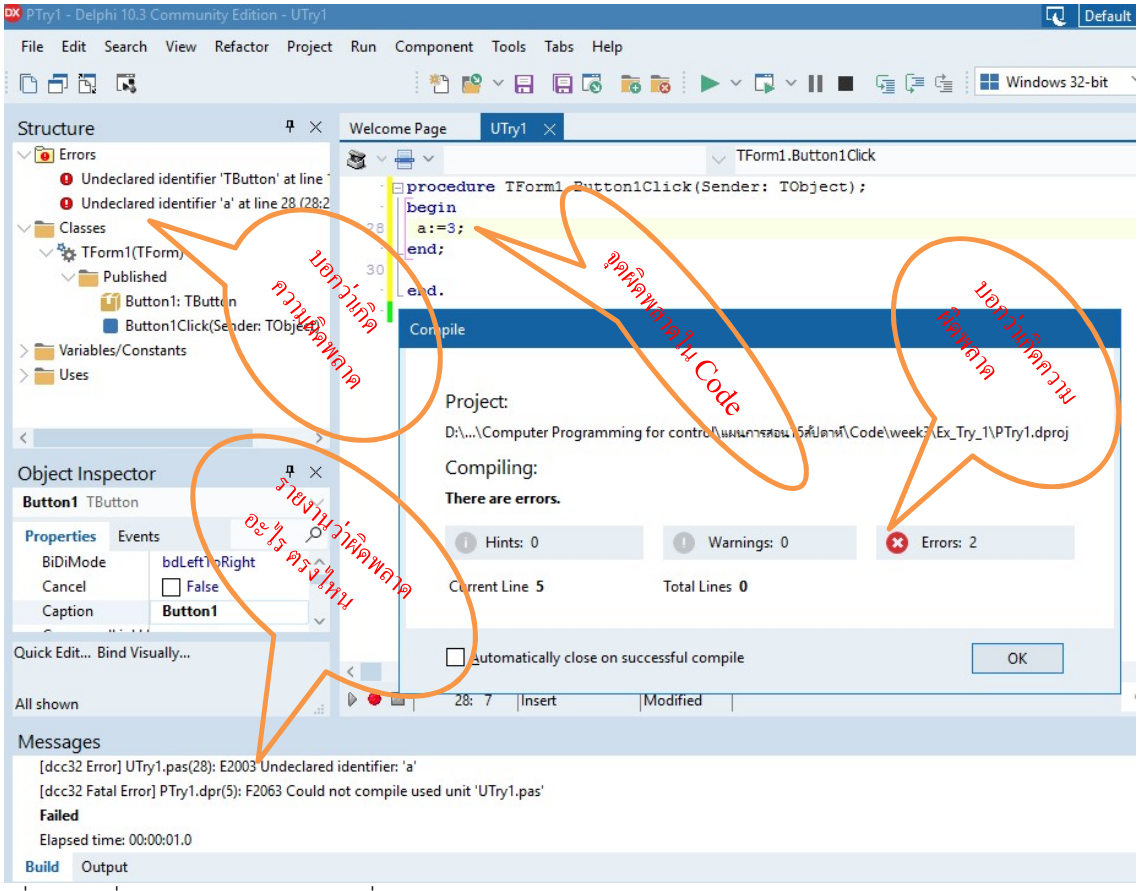
Example

```
//ตัวอย่างการเรียกใช้งาน Function
A:=MyCal();
A:=MyCal;
A:=MyCal(1, 2);
If MyCal(1, 2) = 3 then

//ตัวอย่างการเรียกใช้งาน Procedure
MyWork();
MyWork(1. 2, 3);
```

5. ดักจับ Error ด้วย try except และ try finally

ในการเขียนโปรแกรมนั้นอาจมีข้อผิดพลาดเกิดขึ้นได้ 2 ชนิดใหญ่ๆคือ 1. ผิดพลาดในขณะเขียนโปรแกรม (Design Time) 2. ผิดพลาดหลังจาก Run (Run Time) ข้อผิดพลาดในขณะเขียนโปรแกรมนั้นจะมี Service ของ IDE คอยตรวจสอบอยู่ หรือไม่ก็จะมีการตรวจอีกครั้งตอน Compile Program ถ้าเจอคำสั่งที่ผิดระบบจะไม่ยอมให้ผ่านการ Compile และฟ้องเป็น Report ออกมาว่าผิดอะไรตรงไหน ความผิดพลาดในช่วง Design Time นี้ตรวจจับได้ง่ายและแก้ไขได้ไม่ยาก ในส่วนนี้จะกล่าวถึงวิธีการดักจับข้อผิดพลาดที่เกิดขึ้นหลังจาก Run Program ไปแล้วเท่านั้น



เมื่อเราไม่มั่นใจว่าโปรแกรมในส่วนที่เรากำลังจะเขียนจะมีข้อผิดพลาดจากการใช้งานหรือไม่ เช่น ผู้ใช้งานอาจส่งผ่านค่า 0 มาหารตัวตั้งภายในโปรแกรม (X/0=∞) ทำให้คำตอบเป็นค่า Infinity และเกิด Error ในช่วงการเขียน Code ที่สุ่มเสี่ยงนี้เราสามารถใช้คำสั่ง try คร่อมไว้ดักจับ Error ที่อาจเกิดขึ้นและเตรียมวิธีการแก้ไขไว้ล่วงหน้าได้ การใช้คำสั่ง try มีรูปแบบดังนี้

Format	Details
try-finally	ถ้าเกิดหรือไม่เกิดข้อผิดพลาดจะกระโดดไปทำคำสั่งหลัง final
try-except	ถ้าเกิดข้อผิดพลาดจะกระโดดไปทำคำสั่งหลัง except
try-except on..	ถ้าเกิดข้อผิดพลาดจะกระโดดไปทำคำสั่งหลัง except แยกดักจับข้อผิดพลาดแต่ละชนิดได้

Example

```
// try-finally
Try
   Statement
{Statement...}
```

```
Finally
   Statement
{Statement...}
End;
```

```
// try-except
Try
   Statement
 {Statement...}
 Except
   Statement
 {Statement...}
 End;
```

```
// try except on Exception
Try
   Statement
 {Statement...}
 Except
   On {Name :} Exception type Do Statement
 {Else Statement}
 End;
```

http://www.delphibasics.co.uk/RTL.asp?Name=try

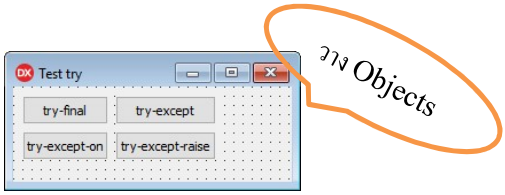
นอกจากจะสามารถดักจับเฉพาะ Error ที่ต้องการได้แล้วยังสามารถใช้คำสั่ง raise สร้าง Exception ขึ้นมาใช้ได้เองอีกด้วย มีรูปแบบดังนี้

Example

```
On E:EZeroDivide do
     raise Exception.Create('Error Div by /0 ');
```

http://www.delphibasics.co.uk/RTL.asp?Name=On

Example




```
procedure TForm1.Bt1Click(Sender: TObject);
var
 a, b:Single;
begin
 try
   b:=0;
   //b:=1;
   a:=3 / b;
   ShowMessage(FloatToStr(a)+'ok');
 finally
  ShowMessage('final');
 end;
end;
```

มาทำที่ finally เสมอ

```
procedure TForm1.Bt2Click(Sender: TObject);
var
 a, b:Single;
begin
 try
   b:=0;
   a:=3 / b;
   ShowMessage(FloatToStr(a)+'ok');
 except
  ShowMessage('final');
 end;
end;

procedure TForm1.Bt3Click(Sender: TObject);
```

Error Div by 0

ถ้าเกิด Error ถึงจะมาทำ

```
var
 a, b:Single;
begin
 try
   b:=0;
   a:=3 / b;
   ShowMessage(FloatToStr(a)+'ok');
 except
  on E:Exception do
   ShowMessage(E.ClassName+' error
message:'+E.Message);
 end;
end;

procedure TForm1.Bt4Click(Sender: TObject);
var
 a, b:Single;
begin
```

ดักทุก Error

```
try
  b:=0;
  a:=3 / b;
  ShowMessage(FloatToStr(a)+'ok');
except
   // Our first exception will not be nmatched
   On E:EInOutError do
     ShowMessage(E.ClassName+' error
message:'+E.Message);
   // This exception will be matched
   On E:EZeroDivide do
     raise Exception.Create('Error Div by /0 ');
   // Catch other errors
   Else
     ShowMessage('Unknown error');
 end;
end;
```

ดักเฉพาะหาร0

สร้าง Exception ขึ้นมาเอง

Code\week3\ Ex_Proc_Event_1

Exercise

- *เขียนโปรแกรมคำนวณ + - * / ที่รับค่า Input 2 ค่ามาจาก TEdit เลือกวิธีคำนวณจาก RadioGroup เมื่อป้อนค่า Input แล้วกดปุ่ม Cal จะไปเรียก Procedure ชื่อ Cal เพื่อเลือกหา Function ที่ต้องการใช้งาน แล้วเรียกใช้ Function นั้นคำนวนว่งคำตอบคืนกลับมาแสดงผลที่ TLabel*
- *เขียนโปรแกรมตรวจสอบการพิมพ์ค่าใน TEdit (Event OnKeyPress) โดยถ้าพิมพ์ ‘a’ ให้สร้าง raise Exception แสดงข้อความว่า ‘Error a!’แต่ถ้าพิมพ์ ‘b’ ให้แปลงค่าเป็นตัวเลขจนเกิด Error ให้ใส่ Try-Except คร่อมไว้ดักจับ Error และแสดงค่า Error Message ออกมา*

Assignment

- *เขียนโปรแกรมที่มีการส่งค่าข้อมูลชนิด TLabel ไปให้ Procedure ทำการเปลี่ยน Property Caption เป็นคำว่า ‘ok’*
- *ทำรายงานชนิดของ Exception ทั้งหมดพร้อมอธิบายความหมายเช่น EzeroDivide คือความผิดพลาดจากการหาร 0 (เขียนด้วยลายมือเท่านั้น)*

Answer Sheet

Code\week3\ Ans_FuncPro_1

Code\week3\ Ans_Try_1

Code\week3\ Ans_Ass_Label1