Volwassenen, Jeugd en Informatica vzw

Schoolstraat 10, 3920 Lommel

Borland Delphi 4.0

voor beginners

Door Stefan Cruysberghs, 1998

Borland Delphi 4.0 voor beginners

Door Stefan Cruysberghs

Inhoudsopgave

Inleiding

In deze cursus gaan we stap voor stap kennis maken met de mogelijkheden van de programmeertaal Delphi.

In het inleidende deel gaan we eerst trachten het principe van programmeren bij te brengen zodat we later kunnen overschakelen naar het principe van object-geörienteerde programma's. De basis van Delphi is Pascal en daarom beginnen we met de belangrijkste Pascal-commando's even op een rijtje te zetten.

Nadien zullen we allerlei kleine voorbeeldjes maken waarmee we de verschillende componenten, funties en procedures zullen leren kennen. Bovendien gaan we algemene programmeer- en Windows-principes bekijken en natuurlijk zullen we ook beter leren werken met de editor en debugger van Delphi.

Deze cursus bevat enkel de voorbeelden en oefeningen. De theorie kan je terugvinden in het handboek of in de Delphi help.

Soms zullen de voorbeelden wat moeilijk lijken. Laat je echter niet afschrikken, na een tijdje zal je merken dat het allemaal kinderspel wordt. Eens je het principe van programmeren begrijpt, zal je merken dat je zeer snel andere commando's of zelfs andere programmeertalen kan leren. Zelf oefenen is eigenlijk het meest belangrijke bij programmeren. Lees dus thuis zeker de cursus tijdig door en probeer eigen voorbeeldjes uit te werken.

De oefeningen en voorbeelden in deze cursus zijn gemaakt met Delphi 4 en 5, maar eigenlijk kan je ze met alle Delphi versies (1 tot 5) gebruiken.

Veel plezier met de geweldige programmeertaal Delphi !!!

Stefan Cruysberghs

Wat is programmeren ?

Programmeren is eigenlijk niets anders dan bevelen geven aan de computer. Een hele reeks van bevelen of commando's na elkaar noemt men een programma.

Er bestaan verscheidene procedurele of hogere programmeertalen zoals Basic, Pascal, C, Assembler, Cobol, Comal, Algol, Fortran, Forth, Modula 2, Logo, ... Tegenwoordig wordt er meestal gewerkt met zogenaamde object-geörienteerde programmeertalen zoals Delphi, Visual Basic, Visual C++, C++Builder, Smalltalk, Powerbuilder, Progress, Java, ... Deze programmeertalen gebruiken nog vaak de commando's van talen zoals Basic, Pascal en C, maar ze beschikken ook over zeer veel mogelijkheden om menu's, dialoogpanelen, vensters, ... aan te maken en je kan met deze talen gemakkelijk databases, foto's, geluid, videobeelden, ... in je programma opnemen.

Bijna alle programmeertalen gebruiken Engelse woorden (=commando's) om de computer bevelen te geven.

Basiselementen in Delphi programma's

Vaste regels

Een Delphi programma bestaat altijd een project-bestand van waaruit de verschillende units gestart worden. Dit project en de standaard unit moeten we zelf niet meer aanmaken. De eerste lessen gaan we enkel met 1 unit (en eigenlijk ook 1 formulier) werken. De basis van een unit ziet er meestal als volgt uit.

```
unit Unit1;

interface

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

type
  TForm1 = class(TForm)
  private
  public
end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

end.
```

Onder **TYPE** staan alle objecten en procedures/functies die je in deze unit gebruikt. Onder **IMPLEMENTATION** begint het eigenlijke programma en dit wordt altijd afgesloten met **END** gevolgt door een punt.

Met de commando's **BEGIN** en **END** kunnen meerdere lijnen samen gehouden worden. Achter elke regel moet je een kommapunt typen, behalve voor de regels die gereserveerde woorden bevatten (var, begin, if, for, repeat, …).

Voor namen van variabelen, functies, procedures, … mag je in een programmeertaal nooit speciale tekens gebruiken en je mag ze nooit laten beginnen met een cijfer.

In Delphi kan je zowel kleine- als hoofdletters door elkaar gebruiken, maar de meeste Delphi-programmeurs gebruiken volgende standaard :

- Alle gereserveerde woorden in kleine letters (= gereserveerde woorden zijn de basiscommando's van een programmeertaal en in Delphi komen deze in het vet te staan)
- Namen van objecten, properties, functies en procedures beginnen met een hoofdletter en indien nodig komen er meerdere hoofdletters in een woord voor. (bv. EersteGetal, IntToStr).
- Wanneer er een begin, if, repeat, … staat wordt er meestal ingesprongen. De meestgebruikte insprong zijn 2 spaties.

Variabelen en typen

a	Naam	Gehuwd
10	Stefan	False

Variabelen kan je voorstellen als de namen voor geheugenplaatsen. In deze variabelen of geheugenplaatsen kan je een bepaalde waarde bewaren. Het toekennen van een waarde kan tijdens het programmaverloop gebeuren, maar het declaren (=het aanmaken) van de variabelen dient in het begin van het programma (globaal) of in de procedure (lokaal) te gebeuren voor het commando **BEGIN** en achter **VAR.**

Variabelen kunnen uit verschillende typen bestaan; gehele getallen, kommagetallen, tekst, … Elke type heeft een andere geheugengrootte. Ook elke property van een component is van een bepaald type. Als je tekst toekent aan een variabele van het type string dient dit tussen aanhalingstekens te staan ' '.

```
var
  a, b, c : Integer;        gehele getallen
  Getal : Real;             kommagetallen
  Naam : String;            tekst
  Gehuwd : Boolean;         waar of onwaar (True, False)
```

Operatoren

Met variabelen of waardes kan je bepaalde berekeningen uitvoeren d.m.v de operatoren. De haakjes werken hetzelfde als in de wiskunde.

Wiskundige operatoren : +, -, *, /
Vergelijkingsoperatoren : =, >, <, <>, =>, =<
Toekenningsoperator : :=

```
a:=10; b:=22;
c:=(a+b)*3;
Getal:=10.365;
Getal:=getal/2;
Naam:='Stefan';
Aanspreking:='Geachte '+Naam;
Ingeschreven:=True;
Aanspreking:=UpperCase(Aanspreking);
Resultaat1:='Het eerst resultaat = '+IntToStr(c);
Resultaat2:='Het tweede resultaat = '+FloatToStr(getal);
Resultaat3:='De sinus van 2 radialen = '+FloatToStr(Sin(2));
```

Indeling in procedures

Alle commando's en functies die je in een Delphi programma wil uitvoeren staan in een procedure. Een procedure is eigenlijk een klein programma op zich. Deze kleine programma's kunnen door elkaar opgeroepen worden. De meeste procedures worden door Delphi zelf aangemaakt en gedeclareerd.

```
procedure Button1Click(Sender : Tobject);
begin

end;
```

Basis commando's

Keuze of selectie : IF ELSE

Een stukje programma wordt alleen uitgevoerd als aan een bepaalde voorwaarde wordt voldaan. Dit noemt men een keuze of selectie. **IF** controleert de voorwaarde en als aan deze voorwaarde niet voldaan is, dan wordt de **ELSE** uitgevoerd. Bij een selectie maakt men gebruik van vergelijkings- of logische operatoren. Met **AND** en **OR** kunnen we voorwaarden combineren. De **BEGIN** en **END** kan je overal in Delphi gebruiken om meerdere lijnen samen te houden. De regel voor de **ELSE** mag jet niet afsluiten met een puntkomma !

```
if a>10 then
begin
  …
end
else
begin
  …
end;
```

```
if (a=10) or (a=20) then
```

```
if a=10 then
begin
  …
end
else
begin
  if a=20 then
  begin
    …
  end
  else
  begin
    …
  end;
end;
```

Herhaling, iteratie of lus : FOR, REPEAT UNTIL

Met het commando **FOR** kan je de computer een reeks commando's meerdere malen laten uitvoeren. Dit noemt men een herhaling, iteratie of lus. We stellen de beginwaarde in met de variabele en met **TO** of **DOWNTO** geven we de eindwaarde op.

```
for i:=1 to 20 do
begin
  …
end;
```

```
for Teller:=5 to 10 do
begin
  …
end;
```

```
for i:=100 downto 0 do
```

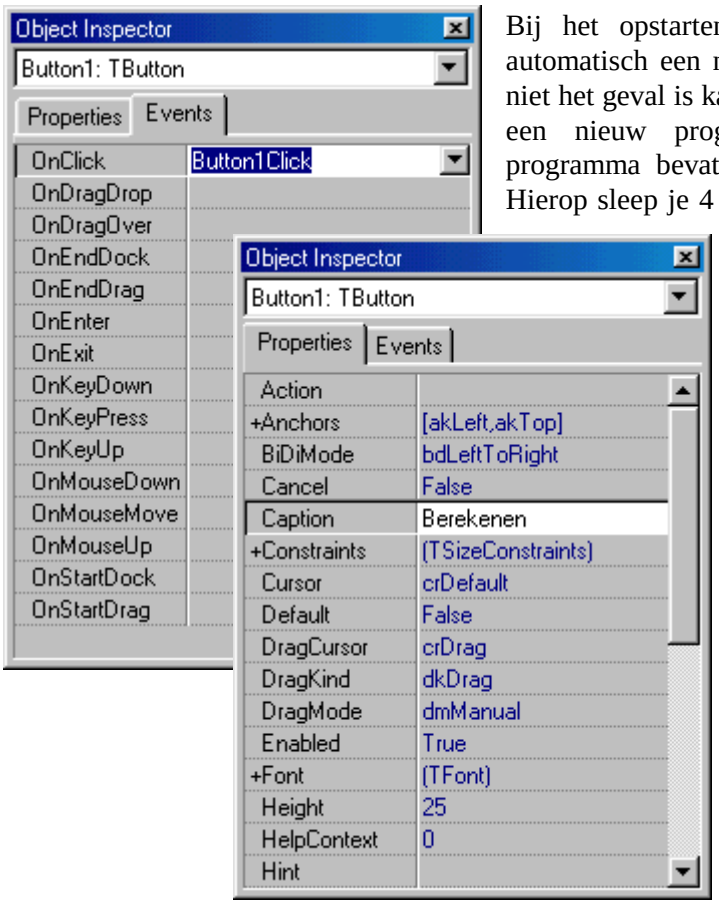
Als je het aantal keren wil laten afhangen van een bepaalde voorwaarde dan moet je de commando's **REPEAT** en **UNTIL** gebruiken. Bij een repeat-until moet je geen begin of end gebruiken om regels samen te houden.

```
repeat
 …
until a=10;
```

```
repeat
 …
until (Gevonden=True) or ((a>10) and (a<20));
```

Voorbeeld

Omdat we in Delphi altijd met een grafische applicatie werken moeten we toch even object-geörienteerd gaan programmeren. We maken als eerste een eenvoudig programma waarmee we de belangrijkste Pascal-commando's kunnen proberen.



Bij het opstarten van Delphi wordt er meestal automatisch een nieuw project aangemaakt. Als dit niet het geval is kan je via **File >> New Application** een nieuw programma starten. Het nieuwe programma bevat automatisch één leeg formulier. Hierop sleep je 4 TEdit-componten en een TButton. Deze kan je terugvinden in het componenten-pallet **Standard**.

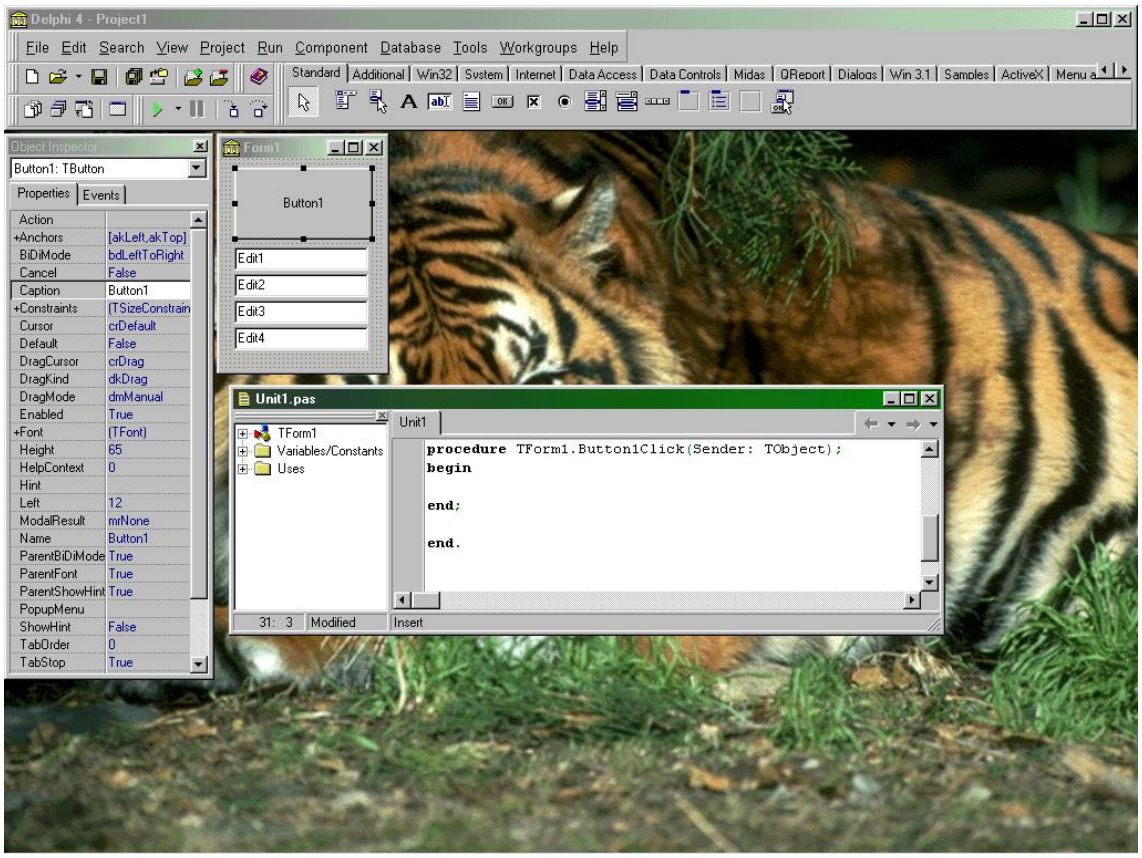
Bij de **OnClick** van de knop gaan we ons eigenlijk programma maken. Eerst dien je de knop aan te klikken en in de **Object Inspector** kan je dan bij het tabblad **Events** naar **OnClick** gaan. In het tabblad **Properties** kan je eventueel de **Caption** (=tekst op knop) veranderen.

Omzetten van datatypes

Omdat de edit-velden enkel een string-waarde kunnen bevatten hebben we nog enkele functies nodig om getallen om te zetten naar strings en omgekeerd.

- **IntToStr(geheel getal)**
- **StrToInt(tekst)**
- **FloatToStr(komma getal)**
- **StrToFloat(string)**

Om de waarde uit een edit-veld te halen of er een waarde in te zetten moeten we de property **Text** gebruiken.



Het eerste voorbeeld zal de gehele getallen, die door de gebruiker in edit1 en edit2 ingetypt worden, optellen en de som in edit3 zetten. Alle getallen gaan we bewaren in een variabele.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Getal1, Getal2 : Integer;
  Som : Integer;
begin
  Getal1:=StrToInt(Edit1.Text);
  Getal2:=StrToInt(Edit2.Text);
  Som := Getal1 + Getal2;
  Edit3.Text:=IntToStr(Som);
end;
```

Uitleg

Allereerst gaan we de variabelen die we nodig hebben declareren. Hiervoor typen we eerst **var** en daaronder alle namen die we willen gebruiken. Achter het dubbelpunt zetten we het type. In dit geval gaan we met gehele getallen werken en het type wordt dus **Integer**. Alle Delphi-regels moet je afsluiten met een kommapunt, behalve de regels met var, begin, if, for, repeat, … Onder de **begin** begint het eigenlijke programma. We gaan eerst de waarden van **Edit1** en **Edit2** opslaan in een variabele. Omdat de **Text**-property van een edit-veld altijd een string is moeten we dit omzetten naar een integer met **StrToInt.** Als beide waardes in **Getal1** en **Getal2** zijn bewaard, kunnen we ermee de som uittellen. Nu we de som hebben moeten we deze in edit3 plaatsen. Dit doen we door de property **Text** van **Edit3** gelijk te stellen aan de variabele **Som**. Deze som, dat een geheel getal is, moeten we wel eerst naar een string omzetten met de functie **IntToStr**.

Oefeningen

Tracht onderstaande oefeningen eerst zelf de maken. Vergelijk nadien jouw programma met de oplossingen. Een programma kan je natuurlijk op verschillende schrijven en daarom staan er bij programma 4, 5 en 6 een hele reeks oplossingen. Tussen haakjes staan de commando's die je zou moeten gebruiken.

1. Schrijf een programma dat 3 gehele getallen optelt en de som in edit4 zet.
2. Schrijf een programma dat 3 kommagetallen vermenigvuldig en de som in edit4 zet.
3. Maak een programma dat de eerste 2 kommagetallen optelt en het derde geheel getal er vanaf trekt. De uitkomt komt ook in edit4 te staan.
4. Schrijf een programma waarbij de gebruiker in het eerste edit-veld een +of een – kan zetten. Nadien wordt het tweede en derde getal opgeteld of afgetrokken al naargelang de waarde van het eerste edit-veld. (**if**)
5. Maak een programma waarbij twee kommagetallen worden ingegeven en na het klikken op de knop komt er in het laatste edit-veld te staan of de getallen gelijk zijn, het eerste getal groter is dan het tweede of het eerste kleiner dan het tweede.
6. Maak een programma dat alle getallen gelegen tussen de waarde in het eerste edit-veld en de waarde in het tweede edit-veld optelt. Vb. 2 tot 5, uitkomst is dan 2+3+4+5. (**for** of eventueel repeat)

Oplossingen

1.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Getal1, Getal2, Getal3 : Integer;
  Som : Integer;
begin
  Getal1:=StrToInt(Edit1.Text);
  Getal2:=StrToInt(Edit2.Text);
  Getal3:=StrToInt(Edit3.Text);
  Som := Getal1 + Getal2 + Getal3;
  Edit4.Text:=IntToStr(Som);
end;
```

2.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Getal1, Getal2, Getal3, Product : Real;
begin
  Getal1:=StrToFloat(Edit1.Text);
  Getal2:=StrToFloat(Edit2.Text);
  Getal3:=StrToFloat(Edit3.Text);
  Product := Getal1 * Getal2 * Getal3;
  Edit4.Text:=FloatToStr(Product);
end;
```

3.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Getal1, Getal2 : Real;
  Getal3 : Integer;
  Uitkomst : Real;
begin
  Getal1:=StrToFloat(Edit1.Text);
  Getal2:=StrToFloat(Edit2.Text);
  Getal3:=StrToInt(Edit3.Text);
  Uitkomst := Getal1 * Getal2 - Getal3;
  Edit4.Text:=FloatToStr(Uitkomst);
end;
```

4.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Getal1, Getal2, Uitkomst : Integer;
begin
  Getal1:=StrToInt(Edit2.Text);
  Getal2:=StrToInt(Edit3.Text);
  if Edit1.Text = '+' then
    Uitkomst := Getal1 + Getal2
  else
    Uitkomst := Getal1 - Getal2;
  Edit4.Text:=IntToStr(Uitkomst);
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Getal1, Getal2, Uitkomst : Integer;
begin
  Getal1:=StrToInt(Edit2.Text);
  Getal2:=StrToInt(Edit3.Text);
  if Edit1.Text = '+' then
    Uitkomst:=Getal1 + Getal2;
  if Edit1.Text = '-' then
    Uitkomst:=Getal1 - Getal2;
  Edit4.Text:=IntToStr(Uitkomst);
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Getal1, Getal2 : Integer;
  Tekst : String;
begin
  Getal1:=StrToInt(Edit2.Text);
  Getal2:=StrToInt(Edit3.Text);
  if Edit1.Text = '+' then
    Tekst:=IntToStr(Getal1 + Getal2);
  else
    Tekst:=IntToStr(Getal1 - Getal2);
  Edit4.Text:=Tekst;
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if Edit1.Text = '+' then
    Edit4.Text:=IntToStr(StrToInt(Edit2.Text) + StrToInt(Edit3.Text))
  else
    Edit4.Text:=IntToStr(StrToInt(Edit2.Text) - StrToInt(Edit3.Text));
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  if Edit1.Text = '+' then
  begin
    Edit4.Text:=IntToStr(StrToInt(Edit2.Text) + StrToInt(Edit3.Text));
  end
  else
  begin
    Edit4.Text:=IntToStr(StrToInt(Edit2.Text) - StrToInt(Edit3.Text));
  end;
end;
```

5.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  KommaGetal1, KommaGetal2 : Real;
begin
  KommaGetal1:=StrToFloat(Edit2.Text);
  KommaGetal2:=StrToFloat(Edit3.Text);
  if KommaGetal1 = KommaGetal2 then
    Edit4.Text:='Getallen zijn gelijk'
  else
    if KommaGetal1 > KommaGetal2 then
      Edit4.Text:='Eerste getal is groter'
    else
      Edit4.Text:='Eerste getal is kleiner';
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  KommaGetal1, KommaGetal2 : Real;
begin
  KommaGetal1:=StrToFloat(Edit2.Text);
  KommaGetal2:=StrToFloat(Edit3.Text);
  if KommaGetal1 = KommaGetal2 then
  begin
    Edit4.Text:='Getallen zijn gelijk';
  end
  else
  begin
    if KommaGetal1 > KommaGetal2 then
    begin
      Edit4.Text:='Eerste getal is groter';
    end
    else
    begin
      Edit4.Text:='Eerste getal is kleiner';
    end;
  end;
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
var
  KommaGetal1, KommaGetal2 : Real;
begin
  KommaGetal1:=StrToFloat(Edit2.Text);
  KommaGetal2:=StrToFloat(Edit3.Text);
  if KommaGetal1 = KommaGetal2 then
    Edit4.Text:='Getallen zijn gelijk';
  if KommaGetal1 > KommaGetal2 then
    Edit4.Text:='Eerste getal is groter';
  if KommaGetal1 < KommaGetal2 then
    Edit4.Text:='Eerste getal is kleiner';
end;
```

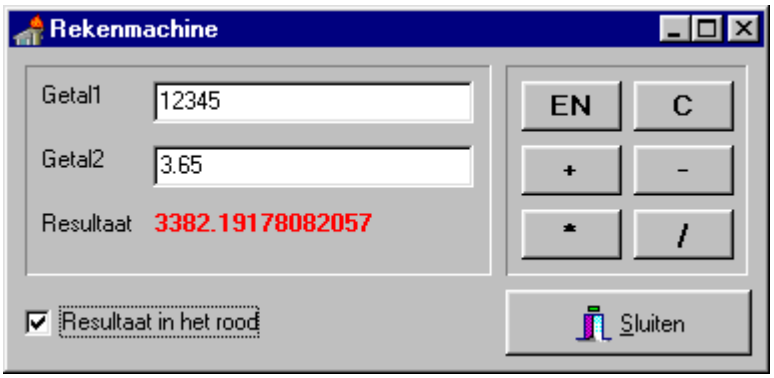
6.

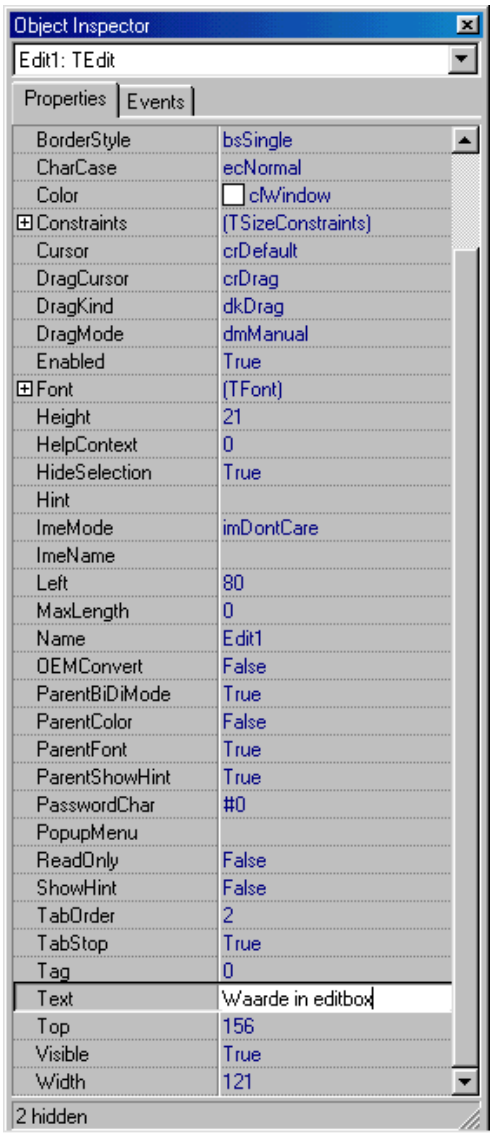
```
procedure TForm1.Button1Click(Sender: TObject);
var
  Getal1, Getal2, Uitkomst, Teller : Integer;
begin
  Getal1:=StrToInt(Edit1.Text);
  Getal2:=StrToInt(Edit2.Text);
  Uitkomst:=0;
  for Teller:=Getal1 to Getal2 do
    Uitkomst:=Uitkomst + Teller;
  Edit3.Text:=IntToStr(Uitkomst);
end;
```

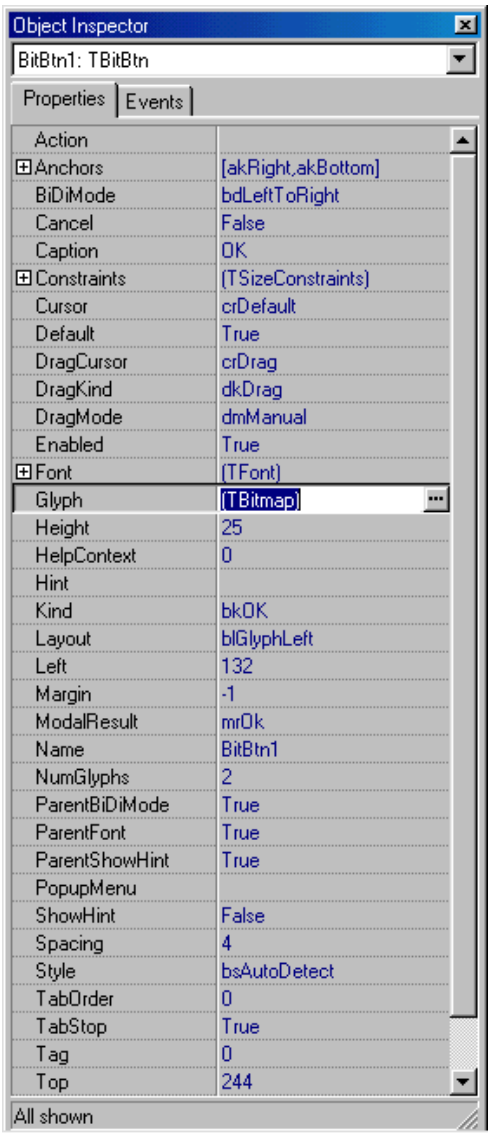
```
procedure TForm1.Button1Click(Sender: TObject);
var
  Getal1, Getal2, Uitkomst, Teller : Integer;
begin
  Getal1:=StrToInt(Edit1.Text);
  Getal2:=StrToInt(Edit2.Text);
  Uitkomst:=0;
  Teller:=Getal1;
  repeat
    Uitkomst:=Uitkomst + Teller;
    Teller:=Teller + 1;
  until Getal1 = Getal2;
  Edit3.Text:=IntToStr(Uitkomst);
end;
```

Rekenmachine

Met dit voorbeeld gaan we trachten de meestgebruikte properties van labels en editboxen onder de knie te krijgen en bovendien gaan we leren werken met variabelen en het omzetten van getallen en strings.







```
unit les2;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, Buttons;

type
  TFormHoofd = class(TForm)
    Edit_Getal1: TEdit;
    Edit_Getal2: TEdit;
    Label_Resultaat: TLabel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Bevel1: TBevel;
    Bevel2: TBevel;
    Button_En: TButton;
    Button_Plus: TButton;
    Button_Maal: TButton;
    BitBtn1: TBitBtn;
    Button_Delen: TButton;
    Button_Min: TButton;
    Button_C: TButton;
    CheckBox_Rood: TCheckBox;
    procedure CheckBox_RoodClick(Sender: TObject);
    procedure Button_EnClick(Sender: TObject);
    procedure Button_CClick(Sender: TObject);
    procedure Button_PlusClick(Sender: TObject);
    procedure Button_MinClick(Sender: TObject);
    procedure Button_MaalClick(Sender: TObject);
    procedure Button_DelenClick(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormHoofd: TFormHoofd;

implementation

{$R *.DFM}

procedure TFormHoofd.CheckBox_RoodClick(Sender: TObject);
begin
  // Kleur van Label aanpassen naargelang waarde Checkbox
  if CheckBox_Rood.Checked=False then
    Label_Resultaat.Font.Color:=clBlack
  else
    Label_Resultaat.Font.Color:=clRed;
end;

procedure TFormHoofd.Button_EnClick(Sender: TObject);
begin
  // Twee strings tegen elkaar plaatsen
  Label_Resultaat.Caption:=Edit_Getal1.Text + Edit_Getal2.Text;
end;

procedure TFormHoofd.Button_CClick(Sender: TObject);
begin
  // Edit-velden en Label leeg maken
  Edit_Getal1.Text:='';
  Edit_Getal2.Text:='';
  Label_Resultaat.Caption:='';
end;
```

```
procedure TFormHoofd.Button_PlusClick(Sender: TObject);
begin
  // Waarden uit Edit-velden van string naar integer omzetten,
  // bij elkaar optellen en als string in Label plaatsen
  // Enkel de invoer van gehele getallen is mogelijk en
  // er worden geen fouten opgevangen
  Label_Resultaat.Caption:=IntToStr(StrToInt(Edit_Getal1.Text)+
    StrToInt(Edit_Getal2.Text));
end;

procedure TFormHoofd.Button_MinClick(Sender: TObject);
begin
  // Aftrekken van de twee gehele getallen
  // Bij foute invoer (letters, kommagetal, ...) verschijnt er geen foutmelding
  // en worden de velden gewist
  try
    Label_Resultaat.Caption:=IntToStr(StrToInt(Edit_Getal1.Text)-
      StrToInt(Edit_Getal2.Text));
  except
    // EConvertError = fout bij omzetten van string naar integer
    on EConvertError do
    begin
      Edit_Getal1.Text:='';
      Edit_Getal2.Text:='';
      Label_Resultaat.Caption:='';
    end;
  end;
end;

procedure TFormHoofd.Button_MaalClick(Sender: TObject);
var
  // integer = geheel getal
  Getal1, Getal2 : integer;
begin
  // Vermenigvuldigen van de twee gehele getallen
  // Elke veld wordt gecontroleerd en enkel het veld met foute invoer wordt gewist
  // Hiervoor worden twee variabelen van het type integer aangemaakt
  Getal1:=0;
  Getal2:=0;

  try
    Getal1:=StrToInt(Edit_Getal1.Text);
  except
    on EConvertError do
      Edit_Getal1.Text:='';
  end;

  try
    Getal2:=StrToInt(Edit_Getal2.Text);
  except
    on EConvertError do
      Edit_Getal2.Text:='';
  end;

  // Als beide getallen groter zijn dan 0, dan pas vermenigvuldigen
  if (Getal1>0) and (Getal2>0) then
    Label_Resultaat.Caption:=IntToStr(Getal1*Getal2)
  else
    Label_Resultaat.Caption:='0';
end;
```

```
procedure TFormHoofd.Button_DelenClick(Sender: TObject);
var
  // Real = kommagetal
  Getal1, Getal2 : real;
begin
  // Delen van de twee kommagetallen
  try
    // String omzetten naar een real (float)
    Getal1:=StrToFloat(Edit_Getal1.Text);
    Getal2:=StrToFloat(Edit_Getal2.Text);
    Label_Resultaat.Caption:=FloatToStr(Getal1/Getal2);
  except
    on EConvertError do
    begin
      Edit_Getal1.Text:='';
      Edit_Getal2.Text:='';
      Label_Resultaat.Caption:='';
    end;
    // EZeroDivide = fout bij delen door 0
    on EZeroDivide do
      Label_Resultaat.Caption:='E';
  end;
end;

procedure TFormHoofd.BitBtn1Click(Sender: TObject);
begin
  // Huidig formulier sluiten
  Close;
end;

end.
```

Tekstverwerker

In dit programmetje gaan we de standaard-componenten gebruiken en kennis maken met hun belangrijkste properties. Het programma is een simpele tekstverwerker waarmee de gebruiker teksten kan ingeven en de layout ervan wijzigen.

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, Menus;

type
  TForm1 = class(TForm)
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    CheckBox3: TCheckBox;
    Memo1: TMemo;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
    SpeedButton3: TSpeedButton;
    BitBtn1: TBitBtn;
    ListBox1: TListBox;
    Button1: TButton;
    Button2: TButton;
    MainMenu1: TMainMenu;
    Bestand1: TMenuItem;
    Linksuitlijnen1: TMenuItem;
    Centreren1: TMenuItem;
    Rechtsuitlijnen1: TMenuItem;
    N1: TMenuItem;
    Afsluiten1: TMenuItem;
    procedure CheckBox1Click(Sender: TObject);
    procedure CheckBox2Click(Sender: TObject);
    procedure CheckBox3Click(Sender: TObject);
    procedure RadioButton1Click(Sender: TObject);
    procedure RadioButton2Click(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
    procedure SpeedButton3Click(Sender: TObject);
    procedure ListBox1Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Linksuitlijnen1Click(Sender: TObject);
    procedure Centreren1Click(Sender: TObject);
    procedure Rechtsuitlijnen1Click(Sender: TObject);
    procedure Afsluiten1Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  if Checkbox1.Checked = True then
    Memo1.Font.Style:=Memo1.Font.Style + [fsBold]
  else
    Memo1.Font.Style:=Memo1.Font.Style - [fsBold];
end;

procedure TForm1.CheckBox2Click(Sender: TObject);
begin
  with Memo1.Font do
  begin
    if Checkbox2.Checked = True then
      Style:=Style + [fsUnderline]
    else
      Style:=Style - [fsUnderline];
  end;
end;
```

```
procedure TForm1.CheckBox3Click(Sender: TObject);
begin
  if Checkbox3.Checked = True then
    Memo1.Font.Style:=Memo1.Font.Style + [fsItalic]
  else
    Memo1.Font.Style:=Memo1.Font.Style - [fsItalic];
end;

procedure TForm1.RadioButton1Click(Sender: TObject);
begin
  Memo1.Font.Color:=clBlack;
end;

procedure TForm1.RadioButton2Click(Sender: TObject);
begin
  Memo1.Font.Color:=clRed;
end;

procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  Memo1.Alignment:=taLeftJustify;
end;

procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
  Memo1.Alignment:=taCenter;
end;

procedure TForm1.SpeedButton3Click(Sender: TObject);
begin
  Memo1.Alignment:=taRightJustify;
end;

procedure TForm1.ListBox1Click(Sender: TObject);
begin
  Memo1.Font.Size:=StrToInt(ListBox1.Items[Listbox1.ItemIndex]);
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  if (Memo1.Alignment = taCenter) and
    (Memo1.Font.Color = clBlack) then
    ShowMessage('De memo voldoet aan de voorwaarde')
  else
    ShowMessage('De memo voldoet NIET aan de voorwaarde');
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  if (fsBold in Memo1.Font.Style) and
    (Memo1.Font.Color = clRed) and
    (Memo1.Font.Size = 14) then
    ShowMessage('De memo voldoet aan de voorwaarde')
  else
    ShowMessage('De memo voldoet NIET aan de voorwaarde');
end;

procedure TForm1.Linksuitlijnen1Click(Sender: TObject);
begin
  Memo1.Alignment:=taLeftJustify;
  Speedbutton1.Down:=True;
end;

procedure TForm1.Centreren1Click(Sender: TObject);
begin
  Memo1.Alignment:=taCenter;
  Speedbutton2.Down:=True;
end;

procedure TForm1.Rechtsuitlijnen1Click(Sender: TObject);
begin
  Memo1.Alignment:=taRightJustify;
  Speedbutton3.Down:=True;
end;
```

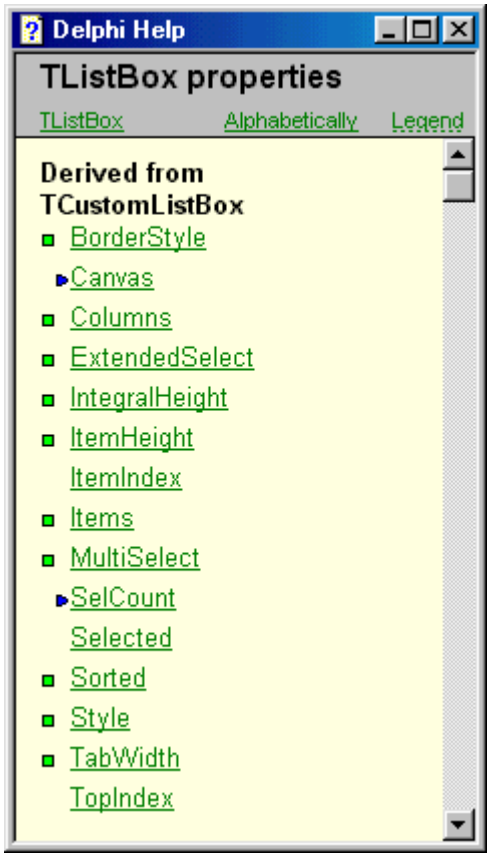
```
procedure TForm1.Afsluiten1Click(Sender: TObject);
begin
  Close;
end;

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

end.
```

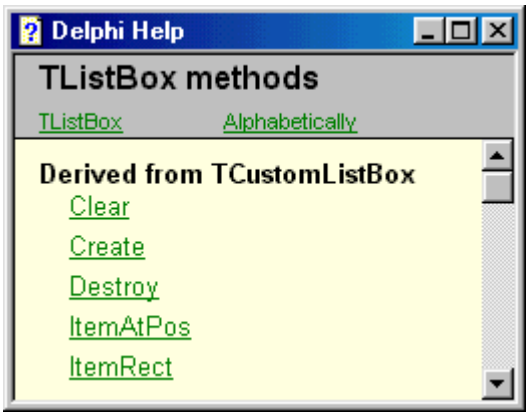
Werken met listboxen

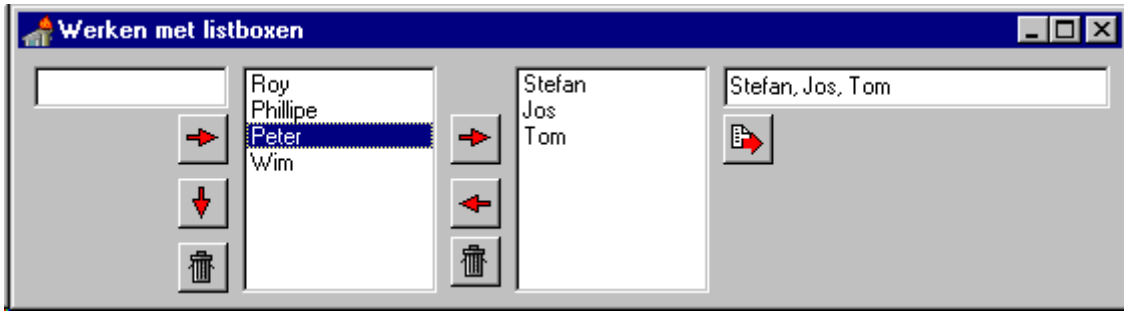
In dit voorbeeldje gaan we werken met listboxen. Lisboxen en comboxen zijn veelgebruikte componenten in Windows programma's. Behalve de properties gaan we ook verschillende methods bekijken die in vele Delphi componenten terug zullen komen.

TListbox (TCombobox)

- **ItemIndex**
- **Items**
  - **Count**
  - **Add(text)**
  - **Move(index1, index2)**
  - **Delete(index)**








```
procedure TForm1.SpeedButton_ToevoegenClick(Sender: TObject);
begin
  Listbox1.Items.Add(Edit1.Text);
end;

procedure TForm1.SpeedButton_Wissen1Click(Sender: TObject);
var
  NrItem : integer;
begin
  NrItem:=Listbox1.ItemIndex;
  if NrItem > -1 then
    Listbox1.Items.Delete(NrItem);
end;

procedure TForm1.SpeedButton_NaarOnderenClick(Sender: TObject);
var
  NrItem, Laatste : integer;
begin
  NrItem:=Listbox1.ItemIndex;
  if NrItem > -1 then
  begin
    Laatste:=Listbox1.Items.Count-1;
    Listbox1.Items.Move(NrItem,Laatste);
  end;
end;

procedure TForm1.SpeedButton_NaarRechtsClick(Sender: TObject);
var
  NrItem : integer;
begin
  NrItem:=Listbox1.ItemIndex;
  if NrItem > -1 then
  begin
    Listbox2.Items.Add(Listbox1.Items[NrItem]);
    Listbox1.Items.Delete(NrItem);
  end;
end;
```

Oefening

Schrijf een klein stukje programma zodat de 3 overgebleven knopjes ook werken :

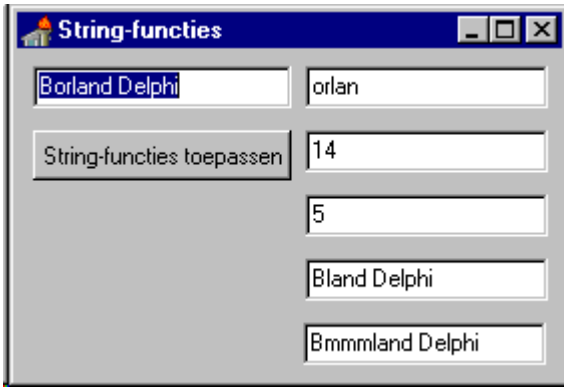
- Element van de rechter lijst naar de linker lijst overzetten
- Element uit de rechter lijst verwijderen
- Alle elementen uit de rechter lijst in edit-veld plaatsen telkens gescheiden door een komma

Functies en procedures

Met volgende kleine voorbeeldjes gaan we kennis maken met een reeks functies en procedures waarmee je strings (teksten) kan bewerken en waarmee je allerlei speciale wiskundige berekeningen kan uitvoeren.

String-functies

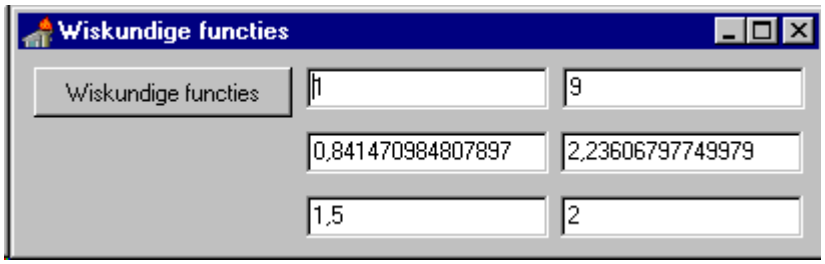
```
function Copy(S: string; Index, Count: Integer): string;
procedure Delete(var S: string; Index, Count:Integer);
procedure Insert(Source: string; var S: string; Index: Integer);
function Length(S: string): Integer;
function Pos(Substr: string; S: string): Integer;
function Uppercase(S : string) : string;
function Lowercase(S : string) : string;
```




```
procedure TForm1.SpeedButton1Click(Sender: TObject);
var
  Tekst : string;
begin
  Tekst:=Edit1.Text;
  Edit2.Text:=Copy(Tekst,2,5);
  Edit3.Text:=IntToStr(Length(Tekst));
  Edit4.Text:=IntToStr(Pos('a',Tekst));
  Delete(Tekst,2,2);
  Edit5.Text:=Tekst;
  Insert('mmm',Tekst,2);
  Edit6.Text:=Tekst;
end;
```

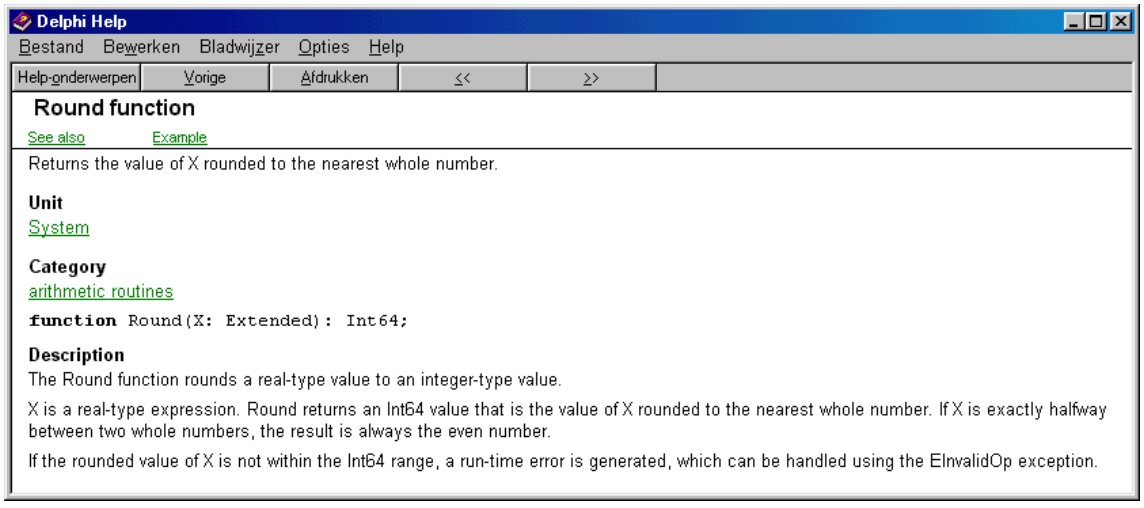
Wiskundige functies

```
function Round(X: Extended): Longint;          Afronden
function Abs(X);                               Absolute waarde
function Cos(X: Extended): Extended;           Cosinus
function Sin(X: Extended): Extended;           Sinus
function Sqr(X: Extended): Extended;           Tweede macht
function Sqrt(X: Extended): Extended;          Vierkantswortel
```




```
Edit1.Text:=IntToStr(Round(1.256));
Edit2.Text:=FloatToStr(Sin(1));
Edit3.Text:=FloatToStr(Abs(-1.5));
Edit4.Text:=IntToStr(Sqr(3));
Edit5.Text:=FloatToStr(Sqrt(5));
Edit6.Text:=IntToStr(Round(Sqrt(Abs(-5))));
```

In de help kan je over al deze functies uitleg en voorbeelden terugvinden.



Grootte-onafhankelijk programmeren

Windows programma's kan je op verschillende resoluties gebruiken (640 x 480, 800 x 600, 1024 x 768) naargelang de grootte van het scherm en de kracht van de videokaart. Een Windows programma moet echter mee vergroten of verkleinen en de gebruiker de mogelijkheid geven de extra ruimte van zijn groot scherm te benutten. Hiervoor kent Delphi verschillende properties en componenten waarmee je grootte-onafhankelijke programma's kan maken.

Property Align

```
None Top, Bottom, Left, Right, Client
```

Met de property Align kan je een panel, memo, … aligneren tegen een bepaalde lijn. Dit kan de zijkant van het form zijn, maar ook een zijkant van een andere panel. Als je top, bottom, left of right kiest zal er altijd een grootte constant blijven. Wanneer je client kiest zal het panel alle vrije plaats innemen.

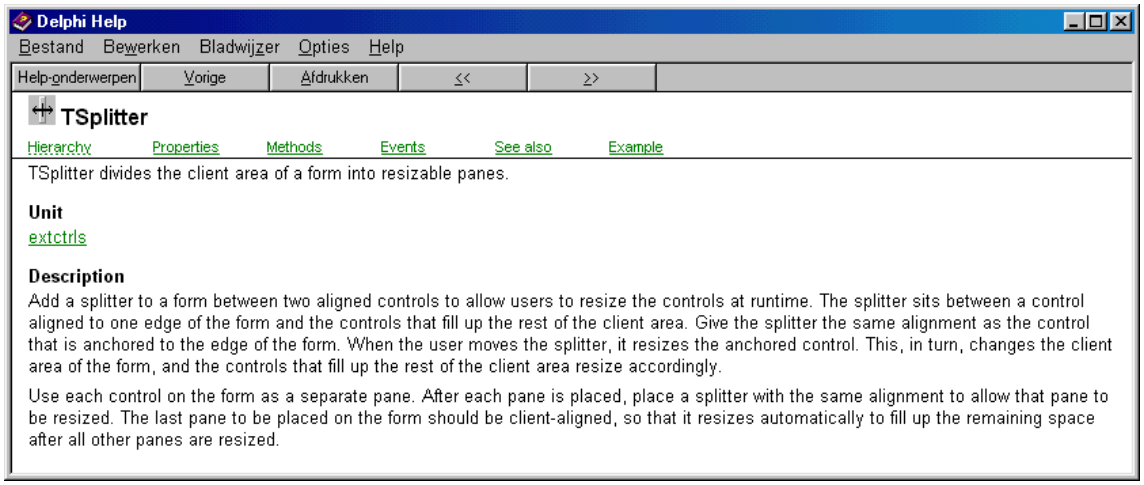
Property Anchors

```
Top, Bottom, Left, Right
```

Met Anchors-properties kan je een component vastankeren op een bepaalde positie. Als je bv. zowel links als rechts instelt, zal het component meevergroten. Als je geen property op True zet zal de relatieve positie behouden blijven.

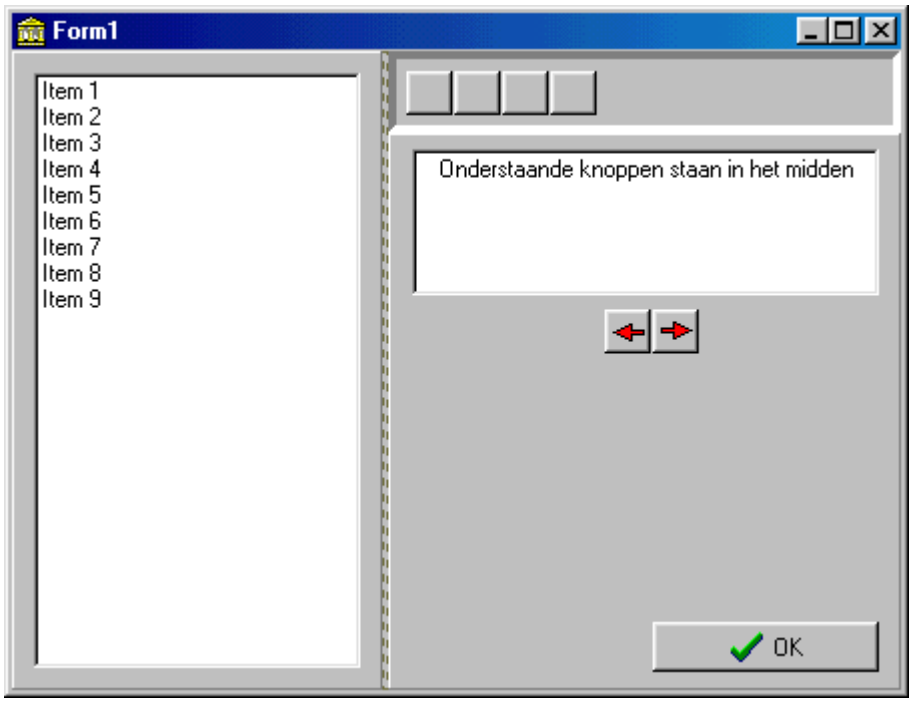
TSplitter component

Een splitter kan je tussen enkele panels zetten en hierdoor kan de gebruiker de grootte van de panels wijzigen.



Oefening

- Het linkerframe kan van grootte gewijzigd worden
- De OK knop moet altijd rechts en onderaan blijven staan
- De memo moet even breed als het form blijven
- De knopjes onder de memo moeten altijd in het midden blijven staan



Grafisch programmeren

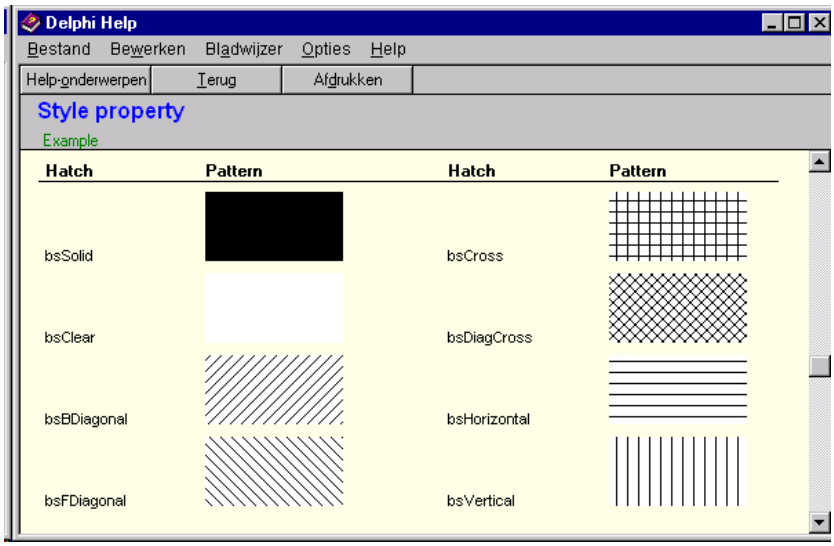
In Delphi kan je tekenen op een **TCanvas** object. Dit object kan je terugvinden bij de meeste grafische objecten zoals **TForm**, **TImage** en **TPaintbox**. In onderstaande voorbeelden gaan we telkens met de canvas van een paintbox werken. De coördinaten die je moet gebruiken hebben als maateenheid **pixels**. De resolutie van het scherm is meestal 640 x 480 pixels, 800 x 600 pixels of 1027 x 768 pixels.

Properties van een Canvas

De belangrijkste **properties** van een canvas zijn :

- **Brush**
- **CopyMode**
- **Font**
- **Pen**

Brush is van het **type TBrush** en heeft op zijn beurt ook weer enkele properties zoals **Color** en **Style**. Een brush is de opvulinstelling van een figuur.



De pen is de instelling van de omtreklijnen. Het is van het **type TPen** die als belangrijkste properties **Color**, **Width, Mode** en **Style** (psSolid, psClear, pmNotXor) heeft.

Methods van een Canvas

De meestgebruikte **methods** bij een canvas zijn :

- **Draw(x,y,graphic)**
- **Ellipse(x1,y1,x2,y2)**
- **LineTo(x,y)**
- **MoveTo(x,y)**
- **Rectangle(x1,y1,x2,y2)**
- **TextOut(x,y,text)**

Voorbeeld

- *Groene rechthoek met dikke rode rand*
- *Blauwe geruite ellips met gele dunne rand*
- *Paarse dikke lijn*
- *Blauwe tekst 'Delphi' met als lettertype Times New Roman en grootte 24*

```
Paintbox1.Canvas.Brush.Color:=clGreen;
Paintbox1.Canvas.Pen.Color:=clRed;
Paintbox1.Canvas.Pen.Width:=5;
Paintbox1.Canvas.Rectangle(10,10,100,100);

Paintbox1.Canvas.Brush.Color:=clBlue;
Paintbox1.Canvas.Brush.Style:=bsDiagCross;
Paintbox1.Canvas.Pen.Color:=clYellow;
Paintbox1.Canvas.Pen.Width:=2;
Paintbox1.Canvas.Ellipse(10,80,200,200);

Paintbox1.Canvas.Pen.Color:=clPurple;
Paintbox1.Canvas.Pen.Width:=6;
Paintbox1.Canvas.MoveTo(0,0);
Paintbox1.Canvas.LineTo(100,200);

Paintbox1.Canvas.Font.Name:='Times New Roman';
Paintbox1.Canvas.Font.Size:=32;
Paintbox1.Canvas.Font.Color:=clBlue;
Paintbox1.Canvas.TextOut(0,100,'Delphi');
```

Kleuren

Om kleuren te gebruiken met je telkens de property Color instellen op een van de vaste constanten (clRed, clGreen, clYellow, ...). Met de functie RGB(r,g,b) kan je zelf een kleur mengen met de 3 hoofdkleuren rood, groen en blauw. De waardes schommelen tussen 0 en 255. Als de 3 kleuren op 0 staan is het zwart en staan ze alledrie op 255 dan is de kleur wit. In een lus kan je op deze manier een of meerdere kleuren systematisch verhogen.

Voorbeeld

```
Paintbox1.Canvas.Brush.Color:=RGB(155,10,200);
```

Voorbeeld

- *Het volgende voorbeeldje tekent een blauwe kleuroverloop. De rechthoek wordt telkens 10 pixels verder geplaats terwijl de blauwe kleur telkens met 7 toeneemt. De omtrekkleur zetten we af door Pen.Style op psClear te zetten.*

```
procedure TForm1.SpeedButton2Click(Sender: TObject);
var
  x, blauw : integer;
begin
  x:=0;
  blauw:=0;
  repeat
    Paintbox1.Canvas.Pen.Style:=psClear;
    Paintbox1.Canvas.Brush.Color:=RGB(0,0,blauw);
    Paintbox1.Canvas.Rectangle(x,0,x+100,100);
    blauw:=blauw+7;
    x:=x+10;
  until x>300;
end;
```

MouseDown, MouseMove en MouseUp events

Verschillende objecten waaronder een paintbox hebben de **MouseDown, MouseMove** en **MouseUp** events. Met MouseDown en MouseUp kan je nagaan of de muisknop is ingedrukt of weer wordt losgelaten. De MouseMove komt voor als je de muis over het object beweegt. In de procedure die bij dit event hoort, krijg je de coördinaten van de huidige muispositie.

Voorbeeld

- *Deze **procedure** tekent een rode cirkel op de plaats waar de muis staat. Verander nadien de kleur eens in RGB(x,y,x)*

```
procedure TForm1.PaintBox1MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
  Paintbox1.Canvas.Brush.Color:=clRed;
  Paintbox1.Canvas.Ellipse(x,y,x+20,y+20);
end;
```

Property Pen Mode

Pen.Mode staat standaard op **pmCopy** waardoor de gewone instellingen gebruikt worden. Als je Pen Mode echter op **pmNotXor** zet kan je ervoor zorgen dat je het vorige figuur terug kan wissen. Onderstaand voorbeeld tekent constant een lijn vanuit punt 0,0 tot aan de muiscursor. De vorige lijn wordt telkens gewist. Hiervoor moet je natuurlijk de vorige positie bijhouden en de lijn opnieuw op die positie tekenen. Omdat **lokale variabelen** die je declareert in de **procedure** telkens terug hun beginwaarde krijgen als **de procedure wordt** opgeroepen, moet je nieuwe **globale variabelen** declareren in het begin van het programma.

Voorbeeld

```
var
  VorigeX, VorigY : integer;

procedure TForm1.PaintBox1MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
  with Paintbox1.Canvas do
  begin
    Pen.Mode:=pmNotXor;
    MoveTo(0,0);
    LineTo(VorigeX,VorigeY);
    MoveTo(0,0);
    LineTo(X,Y);
    VorigeX:=X;
    VorigeY:=Y;
  end;
end;
```

Oefeningen

1. Maak een programma dat een regenboogkleurenoverloop tekent.
2. Maak een programma waarmee een cirkel constant verplaatst kan worden met de muis.
3. Verbeter het vorige programma zodat de cirkel blijft staan wanneer er geklikt wordt. Na het tekenen moet een nieuwe cirkel opnieuw verplaatst kunnen worden.

TTimer component

In het System-pallet vind je een component **TTimer** genaamd. De property **Interval** is de tijd vooraleer het event **OnTimer** uitgevoerd wordt. 1000 is gelijk aan 1 seconde. Bij het event OnTimer zet je het stukje programma dat je om een bepaalde tijd wil uitvoeren.

Dit componentje wordt vaak gebruikt i.p.v lussen. Het nadeel bij lussen (repeat - until, for - next, ...) is dat er ondertussen niets aangeklikt kan worden. Bij een timer werkt de rest van het programma nog gewoon verder.

Met een timer kan je natuurlijk leuke grafische demo's maken en hieronder volgt dan ook een leuk voorbeeldje. Je moet zeker eens proberen de getallen te wijzigen. Automatisch krijg je andere effecten.

Voorbeeld

```
var
  Form1: TForm1;
  straal, r,g,b : integer;

implementation

{$R *.DFM}

procedure TForm1.Timer1Timer(Sender: TObject);
var
  i : real;
  x1,x2,y1,y2 : integer;
begin
  straal:=straal+3;
  if straal>500 then
    straal:=0;

  repeat
    r:=r+16;
    if r>255 then
      r:=0;
    g:=g+13;
    if g>255 then
      g:=0;
    b:=b+10;
    if b>255 then
      b:=0;
    i:=i+pi/65;
    x1:=Round(Sin(i)*straal);
    y1:=Round(Cos(i)*straal);
    x2:=Round(Sin(i)*straal+20);
    y2:=Round(Cos(i)*straal+20);
    Paintbox1.Canvas.Pen.Style:=psclear;
    Paintbox1.Canvas.Brush.Color:=RGB(r,g,b);
    Paintbox1.Canvas.Ellipse(400+x1,300+y1,400+x2,300+y2);
   until i>=2*pi;
end;
```

sin(x)	**sinus**
cos(x)	**cosinus**
round(x)	**afronden van kommagetal (bv. round(5,3)=5)**
pi	**Delphi-constante met waarde 3,1415...**

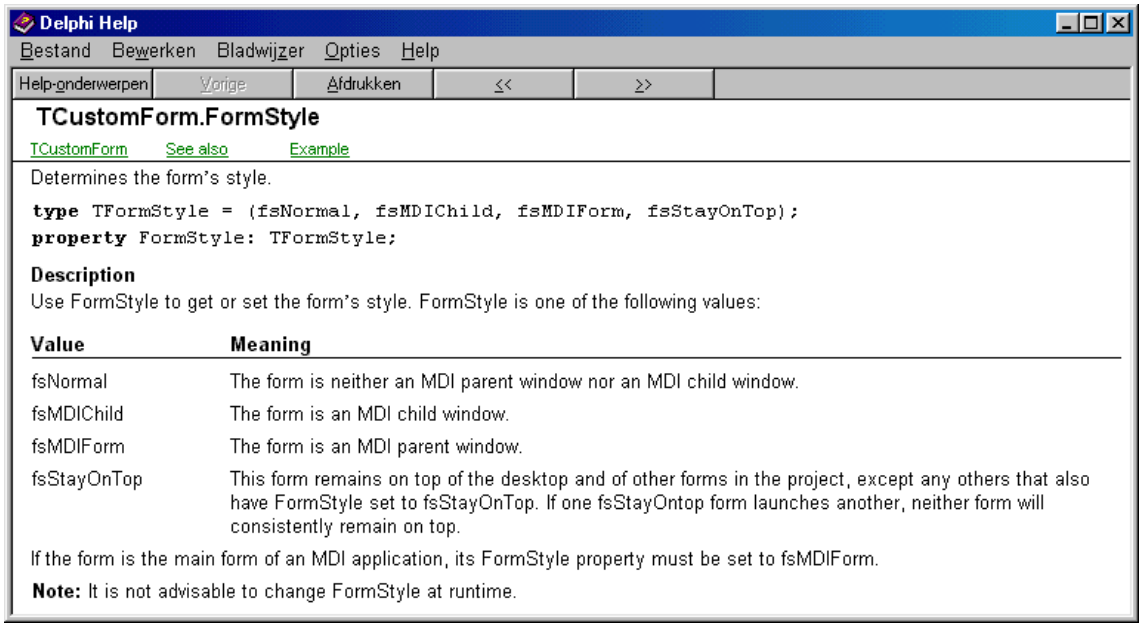
Gebruik van meerdere formulieren

Elke grotere Windows applicatie maakt gebruik van meerdere vensters. Deze vensters kunnen echter op verschillende manieren gebruikt worden. Een modaal venster (vb. venster van foutmelding) dien je eerst af te sluiten alvorens je met de andere venster verder kan gaan. Sommige vensters zijn ook schermvullend maar kan je toch van grootte veranderen terwijl andere vensters een vaste grootte hebben. Al deze mogelijkheden kan je instellen op het form zelf. Met het volgende voorbeeld gaan we alle combinaties eens bekijken.

`TForm`

- `BorderStyle` `bsNone, bsDialog, bsSingle, bsSizeable, bsSizeToolWin, bsToolWindow`
- `FormStyle` `fsNormal, fsMDIForm, fsMDIChild, fsStayOnTop`
- `Position` `poDesigned, poScreenCenter`
- `WindowState` `wsNormal, wsMaximized, wsMinimized`

- `Show` `Formulier tonen`
- `ShowModal` `Formulier modaal tonen. Hierdoor moet het formulier afgesloten worden alvorens de andere formulieren terug gebruikt kunnen worden.`



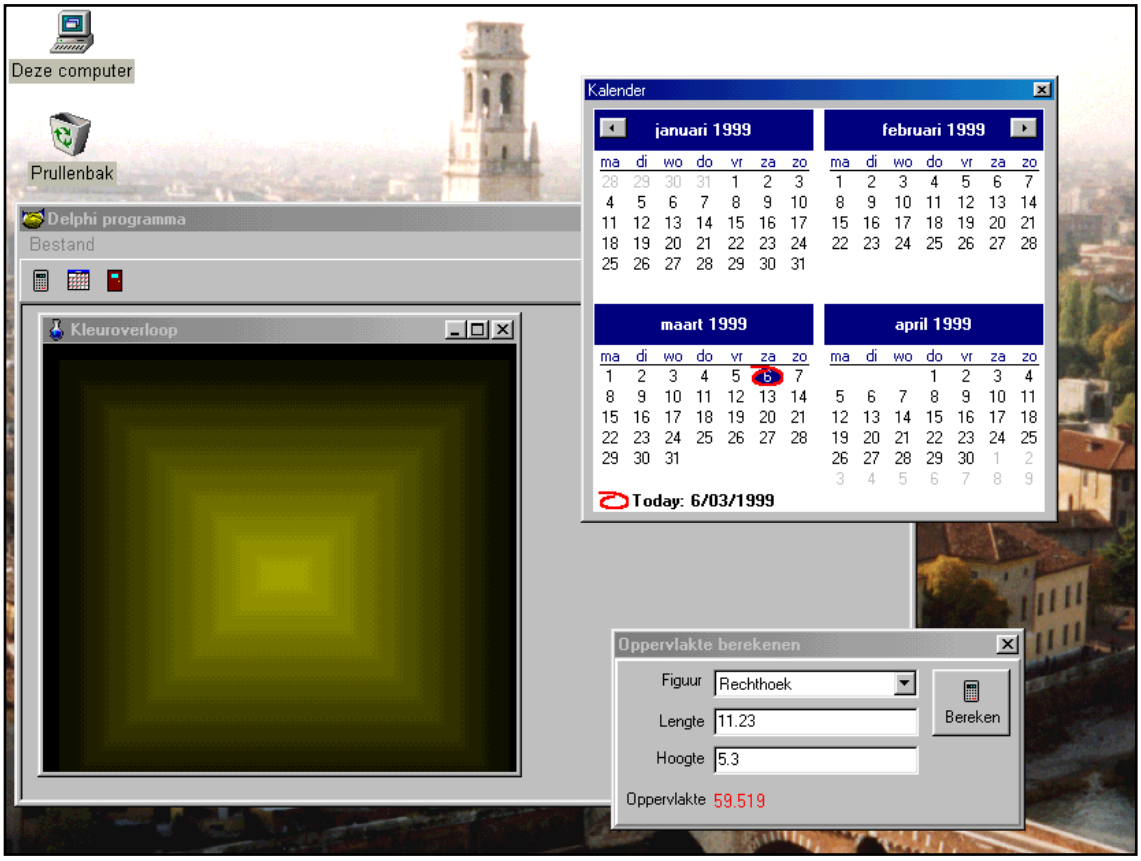
Oefening

Maak een hoofdprogramma met een menu en werkbalk. Vanuit het menu en de werkbalk kan je 2 andere formulieren Kalender en Oppervlakte openen. Het hoofdformulier heeft ook een MDIChild-formulier met daarin een kleuroverloop.

Kalender
Eenvoudig Toolwindow-formulier met kalender-component.

Oppervlakte
Dialoog-form met een combobox, 2 edit-velden, een label en een knop. Zorg ervoor dat het form altijd op de voorgrond blijft, anders lijkt het soms te verdwijnen achter het hoofdformulier. Via de combobox kan gekozen worden uit een rechthoek of gelijkbenige driehoek. De velden zijn de lengte en de hoogte. Na het klikken op de knop worden de waarden gecontroleerd en de gevraagde oppervlakte berekend.

Kleuroverloop
Dit is een MDIChild-formulier met een kleuroverloop. Er worden rechthoeken getekend van de buitenkant naar het midden toe. Bij het verkleinen of vergroten moet de overloop opnieuw getekend worden en er moet dus constant rekening gehouden worden met de hoogte en breedte van dit form. Het tekenen gebeurt rechtstreeks op de canvas van het form.



Oplossing

```
unit Hoofd;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Buttons, ExtCtrls, Menus;

type
  TForm_Hoofd = class(TForm)
    MainMenu1: TMainMenu;
    Bestand1: TMenuItem;
    Afsluiten1: TMenuItem;
    Panel_Werkbalk: TPanel;
    SpeedButton_Oppervl: TSpeedButton;
    SpeedButton_Afsluiten: TSpeedButton;
    N1: TMenuItem;
    Oppervlakteberekenen1: TMenuItem;
    SpeedButton_Kalender: TSpeedButton;
    Kalender1: TMenuItem;
    procedure SpeedButton_OppervlClick(Sender: TObject);
    procedure SpeedButton_AfsluitenClick(Sender: TObject);
    procedure SpeedButton_KalenderClick(Sender: TObject);
    procedure Kalender1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form_Hoofd: TForm_Hoofd;

implementation

// Units van andere forms toevoegen !!
uses kleur, oppervl, kal;

{$R *.DFM}

procedure TForm_Hoofd.SpeedButton_OppervlClick(Sender: TObject);
begin
  Form_Oppervl.Show;
end;

procedure TForm_Hoofd.SpeedButton_AfsluitenClick(Sender: TObject);
begin
  Close;
end;

procedure TForm_Hoofd.SpeedButton_KalenderClick(Sender: TObject);
begin
  Form_Kalender.Show;
end;

procedure TForm_Hoofd.Kalender1Click(Sender: TObject);
begin
  Form_Kalender.Show;
end;

end.
```

```
unit oppervl;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls;

type
  TForm_Oppervl = class(TForm)
    Panel1: TPanel;
    Label5: TLabel;
    Edit_Hoogte: TEdit;
    Label2: TLabel;
    Label1: TLabel;
    Edit_Lengte: TEdit;
    ComboBox_Figuur: TComboBox;
    Label3: TLabel;
    SpeedButton_Bereken: TSpeedButton;
    Label_Oppervlakte: TLabel;
    procedure SpeedButton_BerekenClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form_Oppervl: TForm_Oppervl;

implementation

{$R *.DFM}

procedure TForm_Oppervl.SpeedButton_BerekenClick(Sender: TObject);
var
  Lengte, Hoogte, Oppervlakte : real;
begin
  Lengte:=0;
  Hoogte:=0;
  // Waarden in edit-velden controleren
  try
    Lengte:=StrToFloat(Edit_Lengte.Text);
  except
    on EConvertError do
      Edit_Lengte.Text:='';
  end;

  try
    Hoogte:=StrToFloat(Edit_Hoogte.Text);
  except
    on EConvertError do
      Edit_Hoogte.Text:='';
  end;

  // Als getallen positief en geen fouten
  if (Lengte>0) and (Hoogte>0) then
  begin
    // Aangeklikte waarde in combobox controleren en gepaste berekening maken
    if Combobox_Figuur.Items[Combobox_Figuur.ItemIndex]='Rechthoek' then
      Oppervlakte:=Lengte*Hoogte;
    if Combobox_Figuur.Items.ItemIndex]=2 then
      Oppervlakte:=Lengte*Hoogte/2;
    // Resultaat in label plaatsen
    Label_Oppervlakte.Caption:=FloatToStr(Oppervlakte);
  end;
end;

procedure TForm_Oppervl.FormCreate(Sender: TObject);
// Aanmaken van form
begin
  // Eerste element uit combobox tonen
  Combobox_Figuur.Itemindex:=0;
end;

end.
```

```
unit kleur;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls;

type
  TForm_Kleuroverloop = class(TForm)
    procedure FormResize(Sender: TObject);
    procedure FormPaint(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form_Kleuroverloop: TForm_Kleuroverloop;

implementation

{$R *.DFM}

procedure TForm_Kleuroverloop.FormResize(Sender: TObject);
// Vergroten of verkleinen van form
var
  x, max : integer;
begin
  // Er wordt rechtstreeks op het form getekend dus
  // Height, Width, Canvas, ... is van Form_Kleuroverloop

  // Kleinste waarde van hoogte en breedte nemen
  if Width < Height then
    max:=Width div 2
  else
    max:=Height div 2;

  x:=0;
  // Randkleur afzetten
  Canvas.Pen.Style:=psClear;
  repeat
    Canvas.Brush.Color:=RGB(x,0,0);
    // Rechthoek tekenen van linksboven tot rechtsonder
    // telkens iets meer naar het midden toe
    Canvas.Rectangle(x,x,Width-x,Height-x);
    x:=x+3;
  until x > max;
end;

procedure TForm_Kleuroverloop.FormPaint(Sender: TObject);
// Verplaatsing, overlapping, ... -> hertekenen
begin
  // Procedure FormResize aanroepen
  // Hierdoor hoeft deze procedure niet overgetypt te worden
  FormResize(Sender);
end;

end.
```

Gebruik van de Delphi help-bestanden

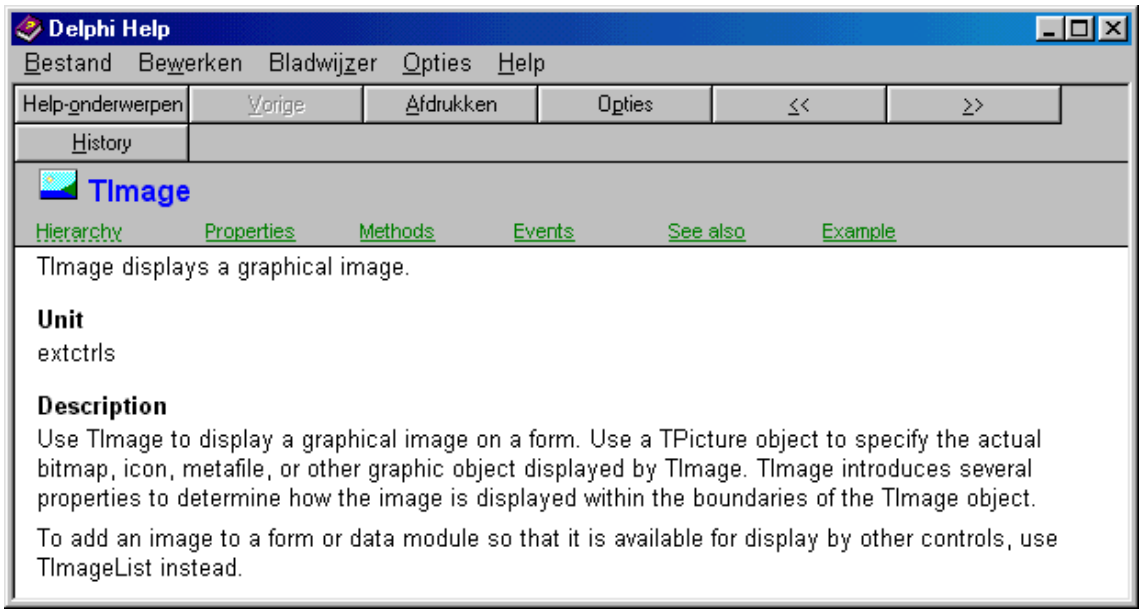
Voorbeeld

Inhoud

Visual Component Library Reference
Categorical Routines Listing
Strings, type conversion and formatting

Index

TImage

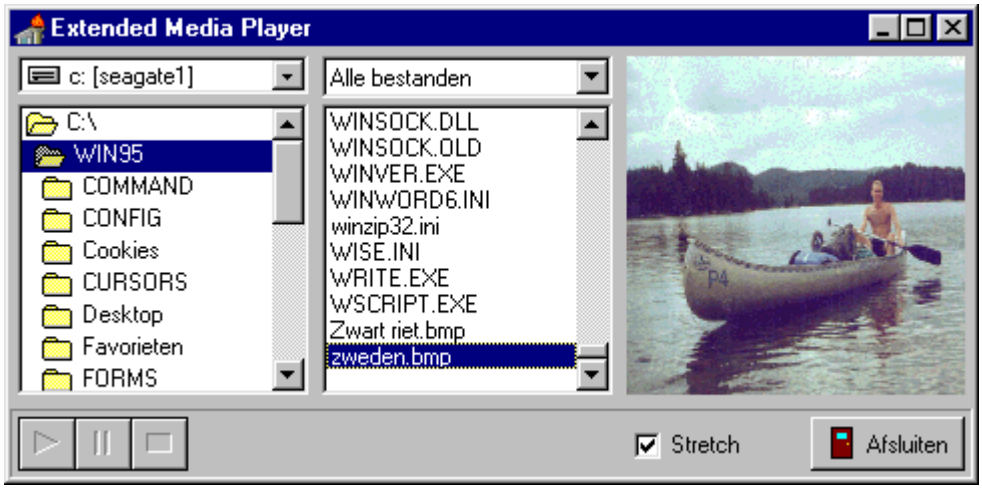


Oefeningen

1. Zoek uit hoe je van enkele Speedbuttons een groep kan maken waarvan maar telkens 1 knop ingedrukt kan zijn ?
2. Hoe kan je een Edit-veld gebruiken om een paswoord in te geven. Hoe kan je er dus voor zorgen dat je niet te zien krijgt wat je aan het intypen bent ?
3. Hoe kan je ervoor zorgen dat een Popupmenu verschijnt wanneer je bv. op een knop klikt. Met welke method kan je een Popupmenu oproepen en op een bepaalde plaats zetten ?
4. Met welke property van een Edit-veld kan je instellen dat de opgegeven tekst niet meer gewijzigd kan worden door de gebruiker ?
5. Met welke functie kan je huidige tijd oproepen en hoe kan je deze tijd omzetten naar een string ?
6. Welke is de exacte waarde van de constante pi ?

Bestanden, directories en multimedia

In het volgende programma gaan we gebruik maken van de componenten om harddisks, directories en bestanden te doorlopen. Bovendien gaan we grafische bestanden en geluiden tonen of afspelen.



Voorbeeld

```
procedure TForm1.FileListBox1Click(Sender: TObject);
var
  Ext : string;
begin
  Cursor:=crHourGlass;
  Ext:=Uppercase(ExtractFileExt(FileListbox1.Filename));
  if (Ext='.BMP') or (Ext='.WMF') then
    Image1.Picture.LoadFromFile(FileListbox1.Filename);
  if (Ext='.AVI') or (Ext='.WAV') or (Ext='.MID') then
  begin
    Mediaplayer1.Filename:=FileListbox1.Filename;
    Mediaplayer1.Open;
    Mediaplayer1.Play;
  end;
  Cursor:=crDefault;
end;

procedure TForm1.CheckBox_StretchClick(Sender: TObject);
begin
  Image1.Stretch:=Checkbox_Stretch.Checked;
end;
```

TDriveComboBox, TFilterComboBox, TDirectoryListBox en TFileListBox component

Deze 4 componenten kan je terugvinden in het componenten-pallet **System**. Via eenvoudige properties kan je ze alle vier met elkaar verbinden.

```
DriveCombobox.DirList      : DirectoryListbox
DirectoryListbox.FileList  : FileListBox
FilterCombobox.FileList    : FileListBox
```

Bij de property **Filter** van de **FilterCombobox** kan je de verschillende bestandstypes en extenties opgeven.

De belangrijkste property van de **DirectoryListbox** is **Directory**. Bij de **FileListbox** is dit de **Filename**. Deze property bevat de aangeklikte bestandsnaam en hier is het uiteindelijk toch allemaal om te doen.

```
ExtractFileExt(Bestandsnaam)
ExtractFilePath(Bestandsnaam)
```

Bovenstaande functies kan je gebruiken om de extentie (punt en 1 of meer karakters) of het pad uit een bestandsnaam te halen. De bestandsnaam bevat namelijk het ganse pad met extentie.

Voorbeeld

```
Extentie:=ExtractFileExt('C:\Bestand\Program\Delphi\Unit1.pas');
```

Extentie = .pas

```
Pad:=ExtractFileExt('C:\Bestand\Program\Delphi\Unit1.pas');
```

Pad = C:\Bestand\Program\Delphi\

Oefening

- Voeg een extra knop toe waarmee je een OpenDialog kan openen zodat een bestand geselecteerd kan worden. Nadien wordt dit bestand ingelezen of afgespeeld en de DirectoryListbox en FileListbox moeten het bestand aan te duiden. Stel hiervoor de properties Directory en Filename in. De Filename van de OpenDialog bevat het ganse pad van het bestand.

TImage component

Zoals bij een Memo kan je bij een Image een tekening (BMP, ICO, WMF) inlezen en bewaren van schijf met de **LoadFromFile** en **SaveToFile** methodes. Deze twee methodes dien je te gebruiken bij de property **Picture**.

Autosize	**automatisch frame aanpassen aan grootte tekening**
Center	**tekening centreren in frame**
Picture	**bitmap (BMP), metafile (WMF) of pictogram (ICO)**
Stretch	**tekening vergroten/verkleinen zodat ze juist in frame past**

TMediaplayer component

De **Mediaplayer** is een zeer knap en krachtig component dat je kan terugvinden in het componenten-pallet **System**.

Met dit component kan je de muziekbestanden midi (MID) en wave (WAV) afspelen, videofilmpjes (AVI) bekijken, CD's beluisteren en zelfs geluid opnemen.

DeviceType	**soort van media-bestand**
EnabledButtons	**actieve knoppen**
Filename	**bestandsnaam**
VisibleButtons	**zichtbare knoppen**
Open	**device openen**
Play	**afspelen**
Pause	**pauzeren**
Eject	**CD-lade openen**
Prev	**naar vorige CD-track springen**
Next	**naar volgende CD-track springen**

Afdrukken met een Canvas

Om in Delphi iets af te drukken kan je het **TPrinter** object gebruiken. Dit object heeft twee belangrijke methods **BeginDoc** en **EndDoc** waarmee je de printopdracht kan starten of stoppen. Voor de rest kan je alles op de canvas van de printer tekenen op dezelfde manier als dit op het scherm gebeurt.

```
Uses Printers;

procedure TForm1.SpeedButton1Click(Sender: TObject);
var
  Rij, Regel : Integer;
begin
  with Printer do
  begin
    BeginDoc;
    Title:='Adruktest Delphi';
    Canvas.Font.Name:='Arial';
    Canvas.Font.Size:=18;
    Canvas.TextOut(200,100,'TITEL');
    Canvas.Font.Name:='Times New Roman';
    Canvas.Font.Size:=12;
    Rij:=300;
    for Regel:=0 to Memo1.Lines.Count-1 do
    begin
      Canvas.TextOut(400,200+Rij,Memo1.Lines[Regel]);
      Rij:=Rij+120;
    end;
    EndDoc;
  end;
end;
```

Slot

Ik hoop dat jullie deze cursus 'Programmeren in Delphi voor beginners' boeiend vonden en verder wil gaan met het programmeren in Delphi of zelfs andere programmeertalen.

Onze vereniging organiseert meestal ook een vervolg cursus voor Delphi waarin nieuwe componenten, andere technieken, … aan bod zullen komen (Extra componenten, Registry, Databases, ….). Bovendien is er tijdens de PC Creatief avonden op vrijdag een Delphi groep actief. Deze mensen trachten regelmatig nieuwe demonstraties te geven en elkaar wat uitleg te geven over allerlei Delphi mogelijkheden.

Op het internet kan je zeer veel informatie, nieuwe voorbeelden en componenten voor Delphi vinden. Hierbij een aantal interessante links:

Borland Benelux
Borland
Borland Delphi Community
Delphi Super Page
Torry's Delphi Page
Delphi 3000
Delphi City
Delphi 32 Information Connection
Delphi Spirit

Delphi Magazine
Delphi Informant

NL Delphi
Dr.Bob [Bob Swart]
Delphi Pagina
Delphi gebruikersgroep
The Delphi Companion

Delphi JEDI Project
GExperts : Open Source Programming Tools

Op mijn eigen website staan ook regelmatige korte artikelen. (http://www.scip.be)

Stefan Cruysberghs
November 1998