Tutorial 1, Delphi: Geldspraak

Versie	Datum	Auteurs	Opmerkingen
1	25-09-2001	Kees Hemerik (code) Roel Vliegen (tekst)	Gebaseerd op Delphi 5 Enterprise.
2	17-03-2005	Kees Hemerik	Aanpassingen: Delphi 7, van gulden naar euro, kleine tekstuele wijzigingen.
3	18-11-2008	Tom Verhoeff	Aangepast voor Lazarus

Inleiding

Doelstelling

Tot nu toe heb je alleen nog maar console-applicaties gemaakt. Het voordeel hiervan is dat ze erg eenvoudig en klein zijn. Een nadeel is dat het er allemaal niet zo mooi uitziet en lastig kan zijn in de bediening. Gelukkig is het tegenwoordig met moderne visuele ontwikkelingstools, zoals bijvoorbeeld Lazarus en Delphi, helemaal niet zo moeilijk om een mooiere interface te maken. Het doel van deze oefening is om een kleine interface te maken bij de unit *Converteer* die een numeriek geld bedrag in spreektaal kan omzetten.

Benodigheden:

- *Lazarus* of een *Delphi* versie, het maakt niet zoveel uit welke. Merk op dat deze oefeningen oorspronkelijk gemaakt zijn met *Delphi 7 Enterprise Edition,* en daarna met *Lazarus* op een Mac. Het kan er op je eigen machine een beetje anders uitzien. Mocht je hier onverhoopt problemen mee hebben, vraag het dan gewoon.
- de unit *Converteer*, die bij deze tutorial wordt meegeleverd.

Geldspraak

Het is de bedoeling dat je in deze oefening een applicatie maakt met twee edit-boxen. In de ene moeten de euro's van een geldbedrag ingevuld kunnen worden en in de andere de centen. Alleen als er een correct bedrag in beide edit-boxen staat moet er op een knop gedrukt kunnen worden, waarna de numerieke representatie wordt omgezet naar spreektaal. Dit resultaat wordt dan in een derde edit-box weergegeven. Deze edit-box moet read-only gemaakt worden.

Oefening Geldspraak

1. Start *Lazarus* op. Je krijgt nu een scherm als in *Figure 1*.

2. Bij het opstarten van *Lazarus* wordt een project voor een GUI applicatie gecreëerd (tenzij de standaard instelling is aangepast; in dat geval creëer je zo'n project door via New Project in het Project menu te kiezen voor Application).

3. Bij aanvang bestaat de GUI applicatie uit het kale *form* (venster) dat in *Figure 1* rechtsonder te zien is. Maak nu een nieuwe map voor dit project en sla het project op met "Save Project As…" uit het project menu. Je zult nu eerst gevraagd worden om een unit op te slaan. Noem deze unit "Main.pas" en sla hem op in de goede map. Nu word je gevraagd om het project een naam te geven. Geef het project de naam "Geldspraak.lpi". De projectnaam is tevens de naam van je applicatie. Je project kan uit meerdere units bestaan, die elk weer eventueel een of meerdere forms kunnen bevatten.

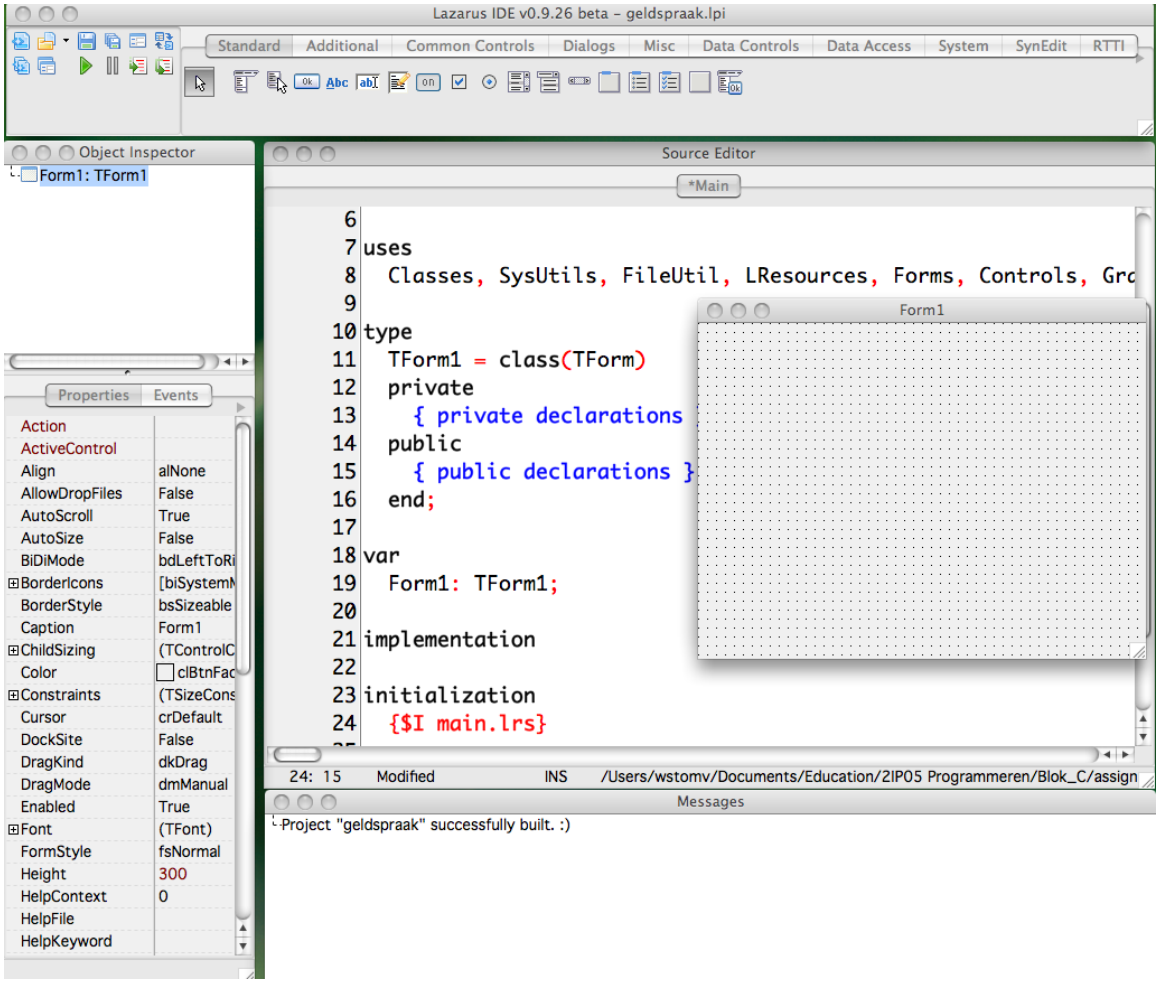

Figure 1

4. Het is van belang om de juiste compiler opties in te stellen. Doe dit via Project > Compiler options. Zie *Figure 2*.

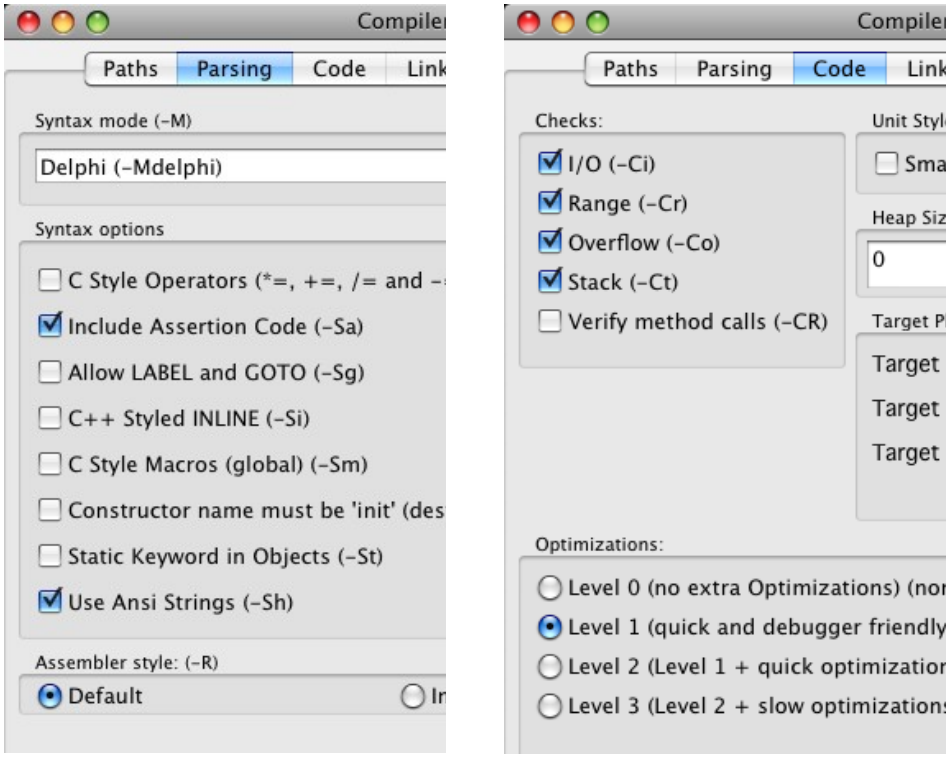

Figure 2

5. Het project dat we net hebben opgeslagen is al een volledig uitvoerbaar project. Je kunt het dan ook al compileren en runnen. Tot nu toe is het slechts een leeg form dat je kunt minimaliseren, maximaliseren, verplaatsen en afsluiten. In feite is dit de kleinste grafische Lazarus applicatie, vergelijkbaar met het standaard Pascal programma:

```
begin
end.
```

Compileer en run het programma eens. Je kunt dit doen door in het Run-menu Run te kiezen of met de “play”–knop in de knoppenbalk. Je kunt het programma beëindigen door het venster af te sluiten.

6. Het form dat je net op beeld zag, wordt beschreven door Pascal code die staat in de unit die je opsloeg in punt 2. Je kunt met *Lazarus* een form op een interactieve grafische manier ontwerpen. Veel aanpassingen die je verricht aan het form zullen automatisch code genereren of een verandering in de code doorvoeren. Verander de naam van het form (“Form1” dus) in “MainForm”. Je kunt dit doen door in de *Object Inspector* (zie *figure 1*) onder Property de Name “MainForm” in te vullen. Let hierbij op de code van de unit, hierin zal automatisch de naam ook worden aangepast.

7. Plaats nu een edit-box op je form. Je kunt dit doen door de component “Edit” te selecteren (van het type TEdit) zoals in *Figure 3*. Klik vervolgens op het form op de plaats waar je wilt dat de edit-box komt te staan. Let hierbij weer op de code die automatisch gegenereerd wordt. Geef de edit-box de naam “EditEuro” en maak hem met behulp van de Text-property leeg.

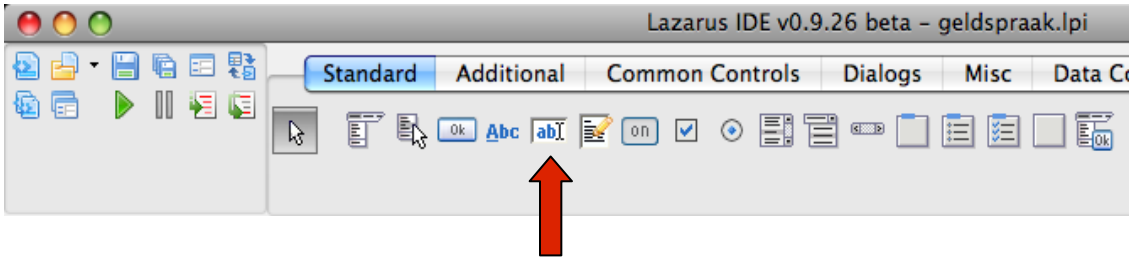

Figure 3

8. Plaats nu een label (in *Figure 3* de **Abc** links naast de edit box) erboven met de naam “LabelEuro” en als caption “&Euro”. Zoals je ziet komt er een streepje onder de “E”, dit betekent dat er nu een sneltoets (<alt-e>) is die naar dit label verwijst. Omdat het natuurlijk niet veel zin heeft als een label de focus zou krijgen, kunnen we met de property FocusControl aangeven welke control de focus moet krijgen in plaats van het label, in dit geval willen we dus dat EditEuro de focus krijgt als we <alt-e> indrukken.

9. Plaats nu op dezelfde manier een edit-box en een label waar de centen van het bedrag in gezet kunnen worden. Geef ook aan deze componenten zinnige namen en koppel er een sneltoets aan. Je hebt nu pas een klein stukje van het programma gemaakt, toch kun je dit nu al uitvoeren. Als je alles goed gedaan hebt, zou je geen foutmeldingen moeten krijgen en krijg je het schermpje dat je hebt gemaakt met de twee edit-boxen. Je kunt er nu al getallen in typen en de sneltoetsen zouden ook al moeten werken.

10. Zet nu een knop (Button) erbij. Ook hier kun je heel makkelijk een sneltoets aan toekennen door in de Caption-property weer een &-tekentje neer te zetten voor de letter die je wilt.

11. De laatste component die nog nodig is is een Edit-box voor het resultaat. Geef deze de naam “EditResultaat”. De gebruiker mag de tekst van deze edit-box niet kunnen aanpassen. Ook dit kun je weer met een property (Read-Only) instellen.

12. Je hebt nu alle componenten op beeld staan die je nodig hebt voor deze oefening. Als het goed is ziet het ongeveer uit als in *Figure 4*. Je hebt nu dus een form waarvan alle standaard functionaliteiten het al doen. De edit-boxen en knoppen functioneren bijvoorbeeld zoals je zou verwachten. Er is alleen nog geen koppeling tussen het form en de code die je erbij zou willen uitvoeren uit de unit *Converteer*. Deze koppeling wordt in interactieve Lazarus-applicaties (en ook de

meeste andere Windows applicaties) gemaakt door het afhandelen van *Events*. Elke component op het form heeft enkele *Events* die kunnen optreden. Zo kan er bijvoorbeeld op de Button geklikt worden, als dit gebeurt dan wordt de *event-handler* voor het *OnClick-event* van de Button uitgevoerd. Als er een wijziging in een edit-box optreedt zal het *OnChange-event* worden uitgevoerd. In de Object Inspector kunnen niet alleen alle Properties worden aangepast, maar ook alle Events. Open in de Object Inspector nu het form en dubbelklik in het Events-tabblad rechts naast *OnCreate* op het vlakje. Er wordt nu automatisch een paar regels code gegenereerd. Voeg aan de body van deze nieuwe procedure de volgende regel toe:

```
Button1.Enabled := False;
```

Het OnCreate-event wordt aangeroepen op het moment dat een object wordt gecreëerd. Door deze regel toe te voegen zal op het moment dat het form gecreëerd wordt de *Enabled-property* van Button1 op false gezet worden, wat tot effect heeft dat er dan niet op de knop gedrukt kan worden.
Dit levert dus de volgende procedure op:

```
procedure TMainForm.FormCreate(Sender: TObject);
begin
  Button1.Enabled := False;
end;
```

Start je programma nog maar eens op en je zult zien dat je niet meer op de knop kunt drukken.

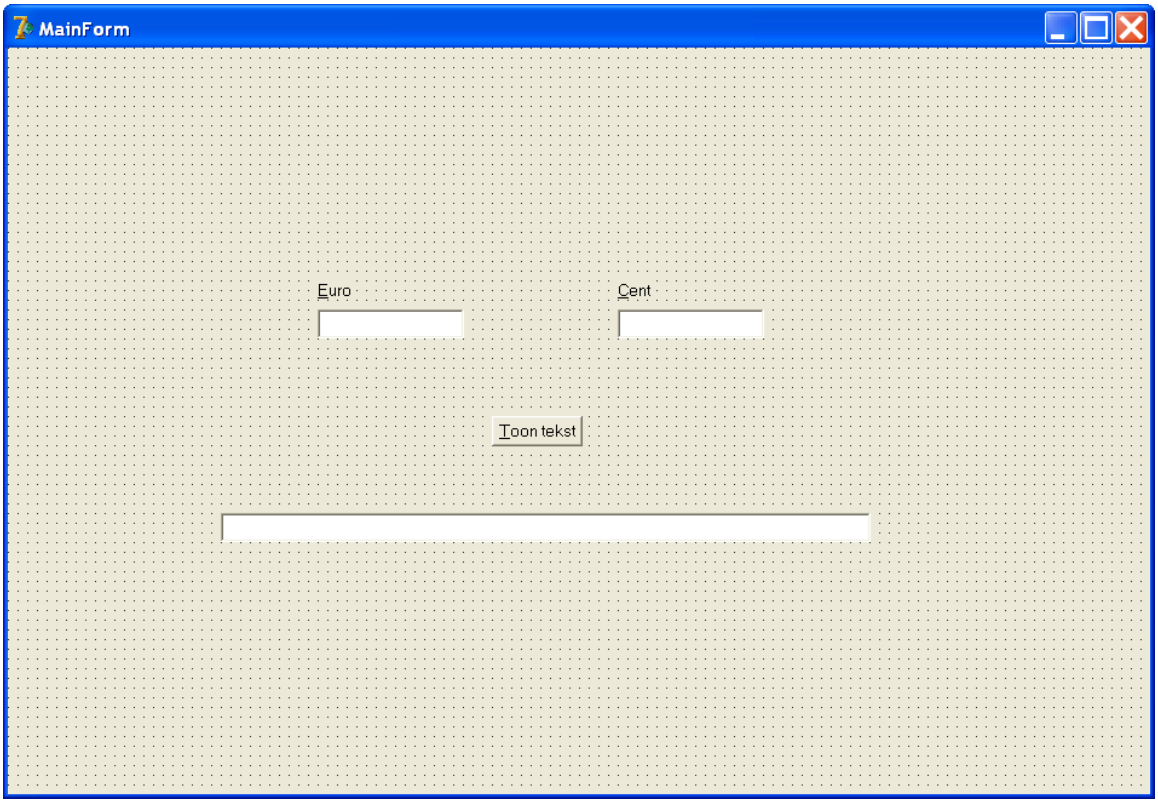

Figure 4

13. Kijk nu maar eens naar de rest van de code die er al staat. Je hoeft nog niet alle code te snappen die er staat, maar zoals je ziet lijkt het erg veel op een normale pascal unit. Om de unit Converteer te kunnen gebruiken, moet deze worden toegevoegd aan je project. Kopieer daarom eerst *Converteer.pas* naar de juiste map. Vervolgens kun je hem toevoegen aan het project door hem te openen (via File > Open) en dan in het Project menu "Add to editor file to project" te kiezen. Om de unit Converteer vervolgens aan te kunnen roepen moet je hem nog achteraan toevoegen achter "uses" in Main.pas, net als bij standaard Pascal:
uses …, …,Converteer;

14. Het is de bedoeling dat de knop alleen gebruikt kan worden indien er geldige getallen zijn ingevoerd in de twee edit-boxen. In de euro-editbox mag een getal staan van ten hoogste 6 cijfers en minimaal 1 cijfer. In de centen-editbox moeten altijd precies 2 cijfers staan. Maak nu een functie ValidEuro die een boolean teruggeeft die aangeeft of de tekst in EditEuro geldig is. Dit doe je door aan de *public declarations* (bovenin de code van de unit in de definitie van TForm1) de functie ValidEuro toe te voegen. Met behulp van "Code Completion" (Ctrl+Shift+C, of door rechter muisknop en "Refactor > Complete Code") kun je aan het implementatie deel automatisch het skelet van de code laten toevoegen, dit gaat bijna hetzelfde als het genereren van een event-handler. Hieronder volgt een complete specificatie en implementatie van ValidEuro:

```
function TMainForm.ValidEuro: Boolean;
{ pre: true
  ret: 1 <= Length(S) <= 6 /\
       (forall i:1<=i<=Length(S). S[i] in Digit),
       waarin S=EditEuro.Text
}
var
  S: String; { tekst van EditEuro }
  I: Integer; { doorloopt S }
begin
  S := EditEuro.Text;
  Result := (1 <= Length(S)) and (Length(S) <= 6);
  if Result then begin
    for I := 1 to Length(S) do begin
      Result := Result and (S[I] in Digit);
    end;
  end;
end;
```

15. Doe hetzelfde nu voor ValidCent:

```
function TMainForm.ValidCent: Boolean;
{ pre: true
  ret: Length(S) = 2) /\ (S[1] in Digit) /\ (S[2] in Digit),
       waarin S=EditCent.Text
```

```
  }
  var
    S: String;
  begin
    S := EditCent.Text;
    Result := (Length(S) = 2) and (S[1] in Digit) and
              (S[2] in Digit);
  end;
```

16. En voor ValidBedrag:

```
function TMainForm.ValidBedrag: Boolean;
{ pre: true
  ret: ValidEuro /\ ValidCent
}
begin
  Result := ValidEuro and ValidCent;
end;
```

17. Als het goed is kan er nu gecontroleerd worden of er een geldig bedrag staat. Als dit zo is moet er op de knop gedrukt kunnen worden. Voeg een OnChange-event toe aan de beide edit-boxen, zodanig dat de knop wordt aangezet (met de enabled-property) indien de functie ValidBedrag true teruggeeft en de knop wordt uitgezet indien de functie false terug geeft.

18. Run je programma nog maar eens. Als je alles goed hebt gedaan zul je zien dat je de knop nu kan gebruiken indien je een correct bedrag invoert. Het enige dat er nu nog moet gebeuren is dat het resultaat in de derde edit-box wordt gezet indien er op de knop gedrukt wordt. Je zou op dezelfde manier als bij 15 naar het OnClick-event kunnen gaan, maar je kunt ook gewoon dubbelklikken op de knop, dan wordt dat event automatisch gekozen omdat het het standaard event is bij een Button. Deze event-handler heeft de volgende specificatie:

```
procedure TMainForm.Button1Click(Sender: TObject);
{ pre: ValidBedrag
  post: EditResultaat bevat tekst bij bedrag in EditEuro
        en EditCent }
```

19. Als het goed is doet je programma het nu. Probeer het nu eens uit te breiden zodat je niet meer op de knop hoeft te drukken om het resultaat op beeld te krijgen als er een geldig bedrag staat.

20. In de meeste windows-programma's kun je door middel van de tab toets de focus wisselen tussen de verschillende componenten op je scherm. Het is prettig als dit in een handige volgorde gebeurt. Deze volgorde kun je instellen met de property TabOrder. Dit is een waarde die uniek is voor alle componenten op je form. De component met de waarde 0 zal na het opstarten de focus hebben. Indien er op tab wordt geduwd zal de component met de volgende TabOrder de focus krijgen.