Agenda für heute, 1. April, 2010

- **Objektorientierte Programmierung**
- Imperatives vs. objektorientiertes Programmieren
- Delphi
- Datentyp: REAL

Lesen Sie den Begleittext Seite 79 – 85

Objektorientierte Programmierung

Eines der drei wichtigsten Programmierparadigmen
(Paradigma: Denkmuster)

Imperative Programmierung

Befehls- oder Anweisungsorientiert

Deklarative Programmierung

Funktionen, Fakten & logische Aussagen (Regeln)

Objektorientierte Programmierung

Objekte mit Eigenschaften und Operationen

Imperative Programmierung

Ein Befehl in der Eingabeaufforderung löst eine ***Folge*** von Anweisungen aus

Befehlszeile

Pascal-Anweisungsfolge

```
C:\TP>fakult 4
Fakultaet von 4: 24

C:\TP>fakult 6
Fakultaet von 6: 720

C:\TP>_
```

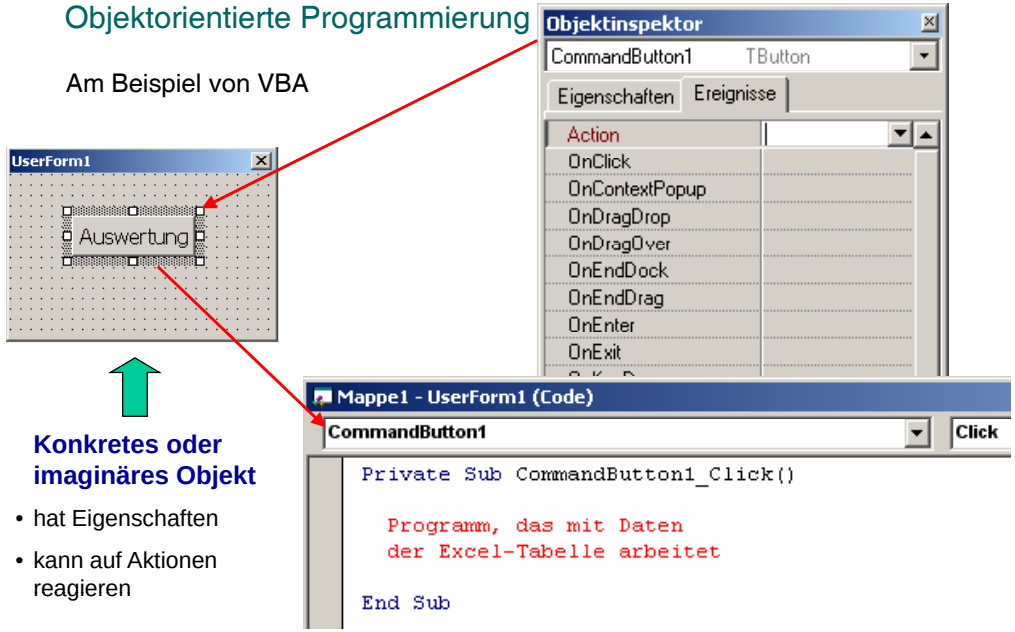
```
program fakult;

var s:string;
    f, i, eingabe, error: integer;

begin
  s:= ParamStr(1);
  val(s,eingabe,error);
  f:= eingabe;
  for i:= eingabe-1 downto 1 do
    f:= f*i;
  writeln('Fakultaet von ',eingabe,': ',f);
end.
```

Bildschirm

Programmcode

Objektorientierte Programmierung

Am Beispiel von VBA



Konkretes oder imaginäres Objekt

- hat Eigenschaften
- kann auf Aktionen reagieren

Objektorientierte Programmierung

Grundidee: Objekte der realen oder imaginären Welt abbilden

- Objekte werden mit Daten und den darauf arbeitenden Prozeduren zu Einheiten zusammengefasst.
- Jedes Objekt kann Botschaften empfangen, Daten verarbeiten und Botschaften senden.
- Ein objektorientiertes Programm kann als eine Menge kooperierender Objekte betrachtet werden.

Wir befassen uns mit konzeptionellen Aspekten der objektorientierten Programmierung um die technischen Grundlagen von Delphi zu verstehen.

Objektorientierte Programmierung

Voraussetzungen für die objektorientierte Programmierung

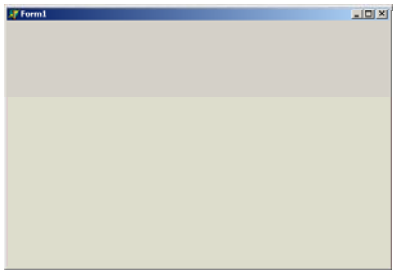
Ereignisse

Ein Programm prüft ständig ob gewisse Ereignisse stattgefunden haben und reagiert darauf mit Aktionen.

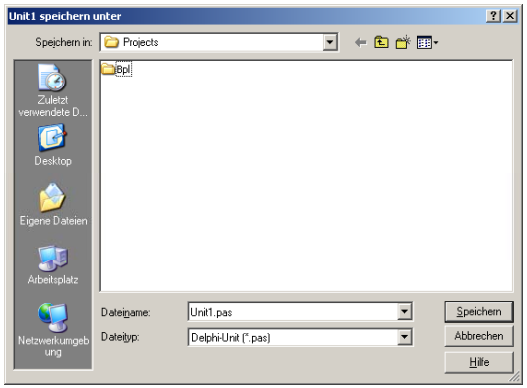
Verwaltung von Objekten

Durch einen Datentyp, in dem sowohl Eigenschaften der Speicherung als auch Prozeduren deklariert sind.

Typische Objekte bei Windows-Anwendungen



System



Typische Ereignisse bei Windows-Anwendungen

Befehlsereignisse

Starten von Programmen, Speichern von Dateien, usw.

Eingabeereignisse

Drücken einer Tastaturtaste, Bewegen der Maus, usw.

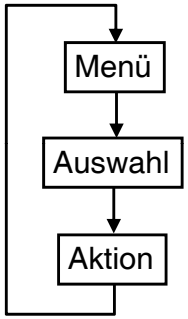
Fensterereignisse

Öffnen, Schliessen, eines Fensters, Grösse verändern, usw.

- Objektorientierte Programmierung
- **Imperatives vs. objektorientiertes Programmieren**
- Delphi
- Datentyp: REAL

Imperatives vs. objektorientiertes Programmieren

Imperativ: Wiederholte, sequentielle Verarbeitung.



Beispiel: zeilenweise Verarbeitung

Objektorientiert: Ereignis-gesteuerte Verarbeitung

Bei Mausklick dividiere

Beispiel: grafische Verarbeitung

Gegenüberstellung

Imperativ

- Daten von Prozeduren getrennt
- Daten zur Manipulation an Unterprogramme übergeben
- Kopieren und anpassen von Prozeduren um ähnliche Probleme zu lösen
- Prozeduren aufrufen und Daten explizit verändern
- Im Vordergrund steht der Lösungsweg

Objektorientiert

- Daten und Methoden vereinigt
- Objekte enthalten Daten, die reagieren können
- Objekte werden vererbt und angepasst um ähnliche Probleme zu lösen
- Objekte erhalten Botschaften auf die sie gezielt reagieren
- Im Vordergrund steht das Problem

- Objektorientierte Programmierung
- Imperatives vs. objektorientiertes Programmieren
- **Delphi**
- Datentyp: REAL

Grundlagen

Object-Pascal

Erweiterung von Pascal durch Elemente der objektorientierten Programmierung:

Klasse	Bauplan für die Abbildung realer Objekte in Softwareobjekte
Instanz	Während der Laufzeit existente, nach dem Bauplan erstellte Objekte
Objekt	Instanz oder auch Klasse & Instanz
Methode	Überbegriff für Prozeduren und Funktionen
Attribut	Beschreiben den Zustand eines Objekts
Eigenschaften	Regeln Lese- und Schreibzugriffe auf Attribute

Grundlagen

Klassendefinition (Bsp.)

```
type
  TAuto = class
  private
    FFarbe: string;
    FBaujahr: integer;
    procedure SetFarbe(Farbe: string);
  public
    property Farbe: string read FFarbe write SetFarbe;
  end;
```

Kann überall dort stehen, wo auch Typ-Deklarationen stehen können

Objektreferenz `var MeinAuto: TAuto;`

Grundlagen

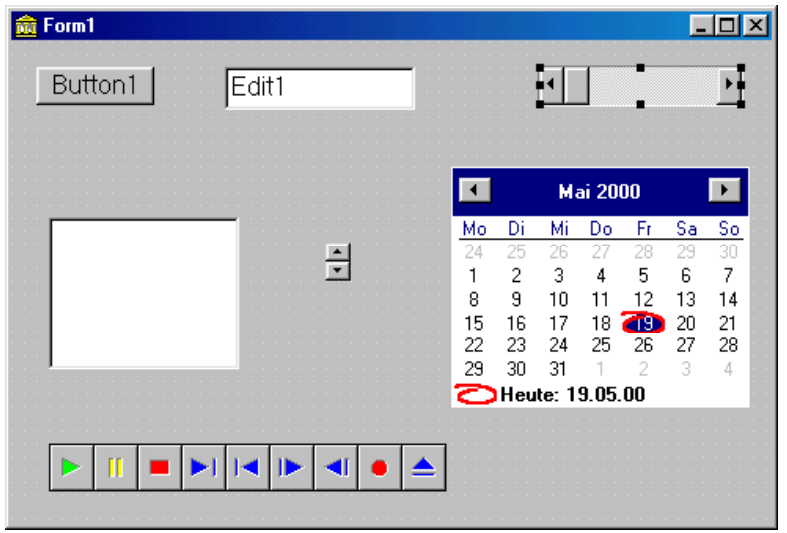
Object-Pascal

Grafische Programmierumgebung

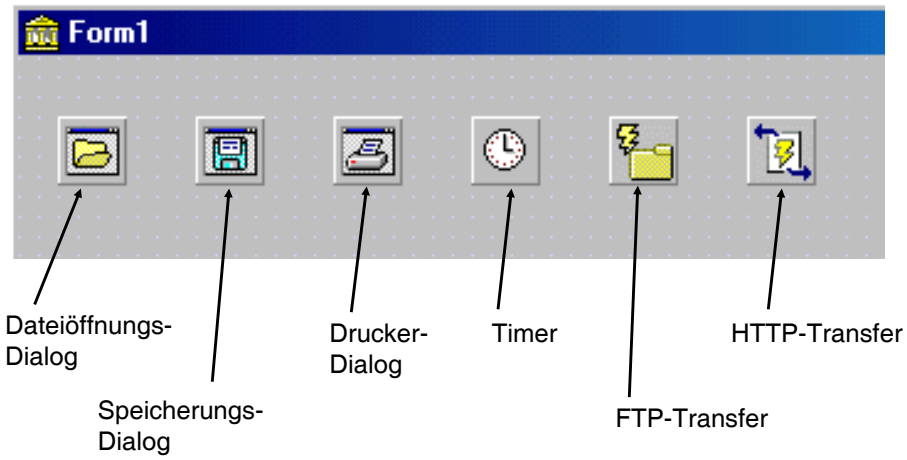
- "Drag-and-drop"-Entwurf von Programmoberflächen
- "Visual Component Library" (VCL)

Integrierte Entwicklungsumgebung (IDE)

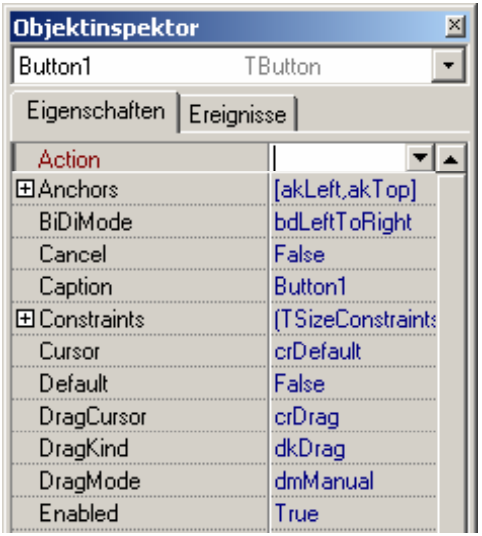
- Verwaltung von Projekten, insb. Abhängigkeiten zwischen Units
- Regeln bez. Programmorganisation

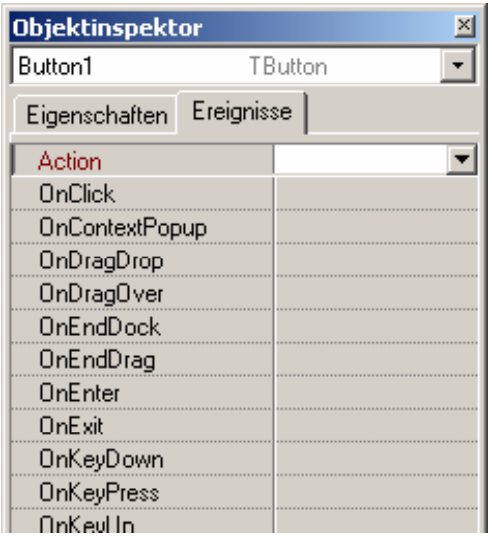


Unsichtbare Objekte in Delphi



Eigenschaften und Ereignisse in Delphi (Bsp.)





Die Projektverwaltung in Delphi

In Delphi wird ein Programm *Projekt* genannt.

Ein Delphi-Projekt besteht aus mehreren Dateien von denen jede durch eine spezifische Erweiterung gekennzeichnet ist:

.dpr	Die Projektverwaltungs-Datei (wird von Delphi automatisch erzeugt und nachgeführt)
.pas	Quelltext-Datei (Objekt-Pascal-Code, in Units gegliedert)
.dfm	Binäre Datei für den Aufbau eines Formulars
.res	Datei mit vom Projekt beanspruchten Ressourcen
.dof	Projektoptionsdatei (Einstellungen für Compiler und Linker)
.exe	Ausführbares Programm

Wichtiger Hinweis: Immer ***alle*** Dateien speichern!!

- Objektorientierte Programmierung
- Imperatives vs. objektorientiertes Programmieren
- Delphi
- **Datentyp: REAL**

Der Datentyp REAL

Teilbereich der reellen Zahlen.

Der Wertebereich ist vom Prozessor abhängig. Beispiel:

5.0×10^{-324} .. 1.7×10^{308} (8 Byte, 15-16 signifikante Stellen)

Operationen: +, –, *, /, ROUND, TRUNC, ABS

Das Resultat eines Ausdrucks mit INTEGER-Werten kann sowohl an eine INTEGER-Variable als auch an eine REAL-Variable zugewiesen werden.

Reelle Typen in Delphi 6

Typ	Bereich	Signifikante Stellen	Grösse in Byte
Real48	2.9×10^{-39} .. 1.7×10^{38}	11-12	6
Single	1.5×10^{-45} .. 3.4×10^{38}	7-8	4
Double	5.0×10^{-324} .. 1.7×10^{308}	15-16	8
Extended	3.6×10^{-4951} .. 1.1×10^{4932}	19-20	10
Comp	$2^{-63}+1$ .. $2^{63}-1$	19-20	8
Currency	922337203685477.5808.. 922337203685477.5807	19-20	8

Der generische Typ REAL ist in der aktuellen Implementation mit dem Typ DOUBLE identisch.

Konsequenzen der endlichen Darstellung für REAL

Gegeben sind folgende Anweisungen:

a:= 4 / 3 – 1;

a:= 3 * a – 1;

Welchen Wert erhält *a*? 3(4/3-1)-1 = 4-3-1 = 0

Angenommen, wir haben 4 Ziffern um eine relle Zahl darzustellen:

4 = 4. 0 0 0

3 = 3. 0 0 0

1 = 1. 0 0 0

das bedeutet:

4/3 = 1. 3 3 3

4/3-1 = 0. 3 3 3

deshalb:

3*(4/3-1) = 0. 9 9 9

3*(4/3-1)-1 = –0. 0 0 1

Also: a = –0. 0 0 1 ≠ 0

Lösung: 3 * 4 / 3 – 3 * 1 rechnen

Automatische Typenkonversion

In Ausdrücken können INTEGER- und REAL-Werte gemeinsam vorkommen.

Wenn für eine der Operationen **+**, **–** oder ***** ein Operand vom Typ REAL ist, dann wird der andere Operand automatisch zu REAL konvertiert bevor der Operator angewandt wird.

Beispiel: (6 + 4) * (1 + 0.1)

Die Faktoren in Klammern werden zuerst ausgewertet:

6 + 4 = 10 (INTEGER)

Im zweiten Faktor ist 0.1 vom Typ REAL, deshalb wird 1 zu REAL konvertiert:

1 + 0.1 = 1.1 (REAL)

Der Multiplikationsoperator * hat einen INTEGER-Operanden (10) und einen REAL-Operanden (1.1). Der INTEGER-Operand wird zu REAL konvertiert:

10.0 * 1.1 = 11.0 (REAL)

Beachte: Das Resultat ist vom Typ "REAL" obwohl es ein ganzzahliger Wert ist!