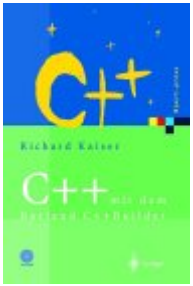
Windows-Programmierung mit C++
Ein Crash-Kurs

Einleitung

Literaturempfehlungen:

• Charles Petzold – Windows-Programmierung (Microsoft Press-Verlag)

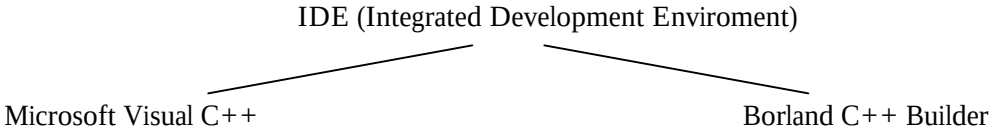


• Richard Kaiser – C++ mit dem Borland C++ Builder (Springer-Verlag)

Bisher: Konsolenprogrammierung (textbasiertes Fenster)
(Anwendung lief als single-task, DOS-Basis)

Jetzt: Windows-Programme (C, C++, C#, Java, VB, Delphi, ...)
(Anwendungen teilen sich Systemressourcen und sind interaktiv)

Was brauchen wir dazu?



! Unter VC und BCB geschriebene Programme sind nicht 100% kompatibel. Bei beiden Vertretern wird visuell programmiert.

Es gibt zwei verschiedene Arten/Konzepte der Vorgehensweise bei der Programmentwicklung in C:

- **C und WinAPI** (application programming interface)
  Sammlung fertiger Klassen in der Master-Header-Datei windows.h; Programmierung erfordert relativ viel „Handarbeit“, fördert aber das Verständnis interner Abläufe und Zusammenhänge)

- **C++ und MFC** (Microsoft foundation classes)
  (rein objektorientierter Entwurf mit speziellen Assistenten, speziell für den Entwurf komplexer Anwendungen geeignet)

! Die WinAPI (auch API oder API32) wird im BCB unter der Bezeichnung VCL (Visual component library) verwaltet. Demzufolge sind die notwendigen Operationen und der Bezeichnung VCL zu suchen!

Es gibt zwei Arten von Programmen:

- SDI (single documents interface)
- MDI (multiple documents interface)

Dabei muss dann noch unterschieden werden, ob z.B. eine Anwendung (exe) oder eine Klassenbibliothek (dll) erstellt werden soll.

Nach dem Kompilieren, Linken und Ausführen mit F9 wird BCB stets mehrere Dateien erzeugen:

mak	Projektdatei
cpp	Quelltext der Unit (Eintrittsdatei mit WinMain-Aufruf)
dfm	Formulardatei
exe	ausführbare Datei
res	Ressourcendatei (Bilder, Sounds)

Hinweise

- Die Projektdatei und der Quelltext dürfen nicht den gleichen Dateinamen bekommen!

- Die Dateien mit der Endung obj, tds , il* und exe können gelöscht werden. Sie werden beim nächsten Start wieder erzeugt! Die mit Tilde ~ beginnenden Dateitypen sind Backupdateien. Hierzu ist ex ratsam sich eine .bat-Datei anzulegen mit folgendem Inhalt:

  ```
  del *.~*
  del *.obj
  del *.tds
  del *.il?
  ```

Nach ihrem Aufruf werden die unnötigen Dateien automatisch gelöscht.

- Jedes weitere Formular erzeugt eine .cpp, eine .dfm und eine .h-Datei!

- Eine vom C++Builder automatisch erzeugte Funktion sollte man nicht manuell löschen. Der C++Builder entfernt eine solche Funktion beim nächsten Kompilieren automatisch, wenn zwischen den geschweiften Klammern kein Text steht.

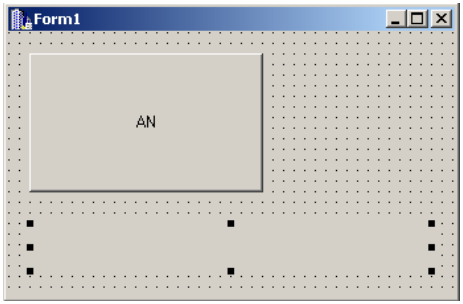
- Interessant auch die Tastenkombination STRG+SHIFT+I (bzw. U) zum Ein-bzw. Ausrücken markierter Textblöcke!

- In BCB existiert eine kontextsensitive Hilfe, die nach dem Anklicken der zu hinterfragenden Komponente/Anweisung mittels F1 abgerufen werden kann.

Ein erstes Projekt (versuch0proj.mak)

```
void __fastcall TForm1::Knopf1Click(TObject *Sender)
{
  if (Label1->Caption=="")
   Label1->Caption = "HUHU HIER BIN ICH!";
  else Label1->Caption = "";

  if (Knopf1->Caption == "AN")
   Knopf1->Caption = "AUS";
  else Knopf1->Caption = "AN";
}
```



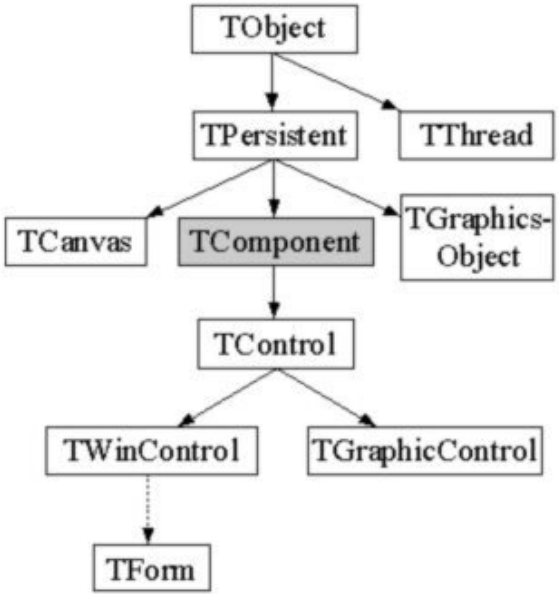
Benötigt wird ein Button (Knopf1) und ein Label (Label1).

Hier wird demonstriert, wie man Attribute manipuliert und wie man Ereignissen (hier: Knopf1Click) eine bestimmte Aktion (Methode) zuordnet.

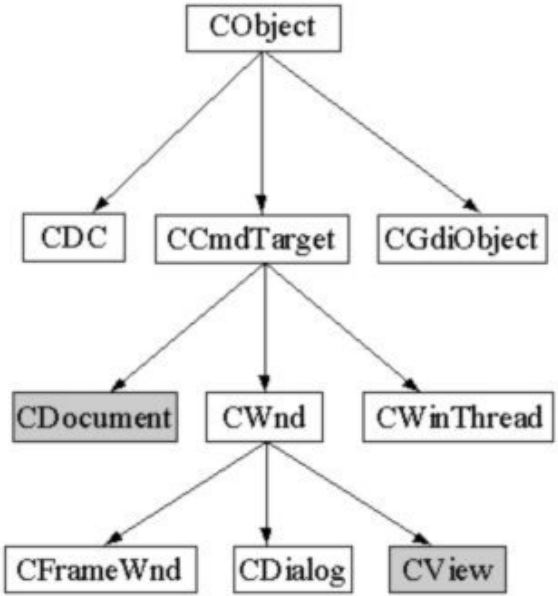
(Wenn auf den Button geklickt wird und das Label keinen Text enthält, bekommt es einen Text, sonst wird sein Text gelöscht. Außerdem wird die Beschriftung (Caption) des Buttons gewechselt)

Die Klassen der VCL

Vergleich VCL vs. MFC

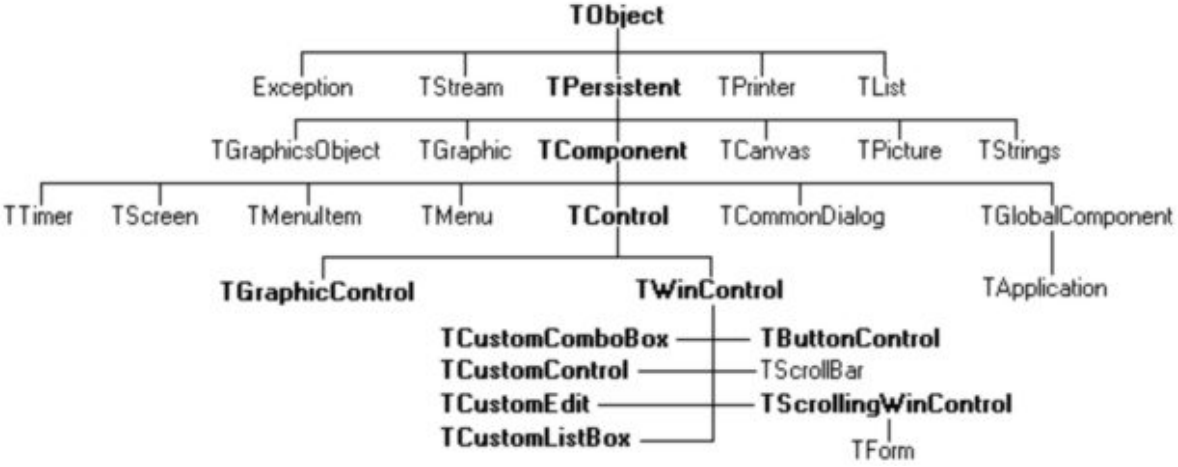


VCL-Architektur



MFC-Architektur

Hierarchie innerhalb der VCL (Ausschnitt)



Die wichtigsten Komponenten eines Formulars: (Komponenten=Bausteine, das Formular ist auch eine Komponente!)

Komponente	Ziel	Wichtige Attribute
 Label	zeigt Text auf Formular an, auch zur Ausgabe	AutoSize, Caption, Visible, WordWrap
 Edit	einzeiliges Eingabefeld, dient auch für Ausgaben; Markierung und Zwischenablage möglich; Inhalte sind Strings!!	AutoSelect, ReadOnly, Text (kann auch leer sein)
 Button	Schaltfläche zur Ausführung von Aktionen (BitButton kann zusätzlich Grafiken aufnehmen)	Caption, Name
 RadioButton	Optionsfelder, Auswahl verschiedener Möglichkeiten (beachte: teilweise Gruppierung mittels GrouBox oder RadioGroupBox nötig)	Caption, Checked
 CheckBox	Kontrollfeld, stellt Option bereit, diese kann aus und angeschalten werden	Caption, Checked

 GroupBox, RadioGroupBox	Windows-Gruppenfeld, wenn ein Steuerelement innerhalb einer Gruppe platziert wird, ist das Gruppenfeld das übergeordnete Objekt dieser Komponente.	Caption, (Columns, Items zus. bei RadioGroupBox)
 Memo	mehrzeiliges Eingabefeld	Lines, ScrollBars, WantReturns (Enter zum Zeilenwechsel erlaubt)
 ListBox	Listenfeld für Auswahl von Optionen	Columns, Items, MultiSelect, Sorted
 ComboBox	Kombination aus einem Eingabefeld und einem bildlauffähigen Listenfeld (Auswahl eines Elementes)	DropDownCount, Items, Sorted
 MainMenu	Hauptmenü mit Menüleiste	Items
 PopupMenu	erscheint, wenn der Benutzer die Komponente markiert und mit der rechten Maustaste klickt	Items

Alle Komponenten habe das Attribut „Name“, welches die Bezeichnung für das Objekt darstellt. Neben den Attributen können jeder Komponente auch Ereignisse (Events) zugeordnet werden.

Alle oben genannten Komponenten sind visuelle Komponenten. Darüber hinaus gibt es auch nicht-visuelle Komponenten (z.B. Timer, Open, Save, ...). Diese erzeugen kein sichtbares Objekt während der Laufzeit, sondern dienen Steuerfunktionen.

Behandeln von Laufzeit-Fehlern

```
try
{
  //kritische Anweisungen (z.B. Öffnen einer Datei, Typkonvertierungen, ...);
}
catch(...)          //...bedeutet: bei jeder Art von Fehler
{
  //Fehlerhandlung
}
```

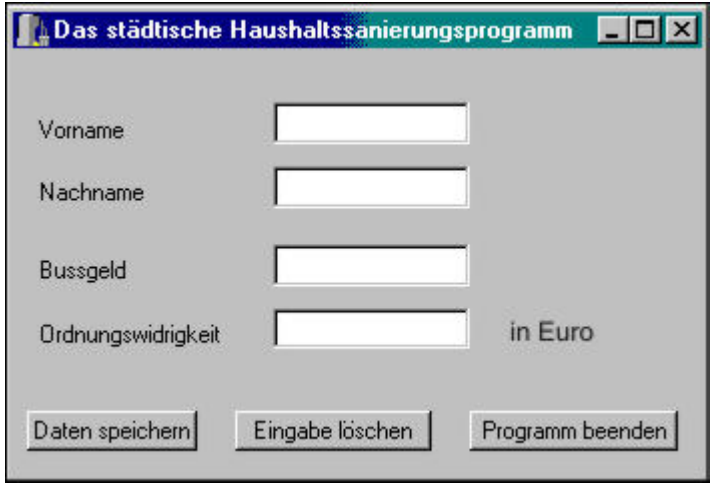
Beispiel:

```
try
{
  Memo1->Lines->LoadFromFile(„Geruest.cpp“);
}
catch(Exception *Ausnahme)   //Objekt der Klasse Exception
{
  ShowMessage(Ausnahme->Message);
}
```

Aufgaben

Aufgabe 1

1. Schreiben Sie ein Programm, das ein Fenster mit folgenden Elementen anzeigt:



Verwenden Sie dazu die Komponenten Label, Edit und Button von der Seite Standard der Komponentenpalette.

2. Ersetzen Sie alle vom C++Builder vergebenen Namen durch aussagekräftige Namen. Da in diesem Beispiel sowohl ein Label als auch ein Edit-Fenster für den Vornamen, Nachnamen usw. verwendet wird, kann es sinnvoll sein, den Typ der Komponente im Namen zu berücksichtigen, z.B. LVorname und LNachname für die Label und EVorname und ENachname für die Edit-Fenster.

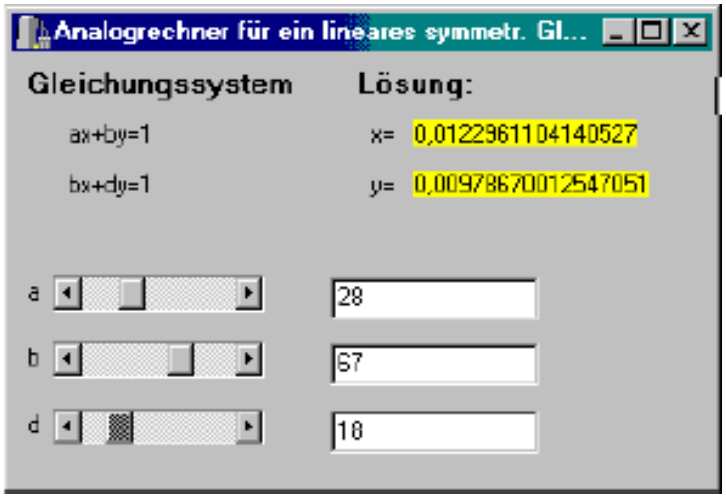
3. Als Reaktion auf ein Anklicken des Buttons Eingabe löschen soll jedes Eingabefeld mit der Methode Clear gelöscht werden. Für den Button Daten speichern soll keine weitere Reaktion vorgesehen werden. Beim Anklicken des Buttons Programm beenden soll das Formular durch den Aufruf der Methode Close geschlossen werden.

Aufgabe 2

Schreiben Sie folgendes Programm:

Der „Wonder“-Button soll bei Klick auf den „Kleiner um 2“-Button um 2 Pixel in Höhe und Breite verkleinert werden, wenn er mindestens die Länge und Höhe 3 hat. Der „Reset“-Button soll ihn wieder auf Originalmaß bringen (der „Wonder“-Button selbst schließt die Anwendung).

Aufgabe 3


Analogrechner für ein lineares symmetr. Gl...
Gleichungssystem
Lösung:
ax+by=1
bx+dy=1
x= 0,0122961104140527
y= 0,0097867001254705
a
b
d
28
67
18

Schreiben Sie ein Programm zur Lösung des symmetrischen linearen Gleichungssystems
$ax + by = 1$
$bx + dy = 1$
bei dem man die Koeffizienten a, b und d an Schiebereglern einstellen kann.

Diese Koeffizienten sowie die Ergebnisse sollen digital in einem Edit-Feld dargestellt werden, wenn einer der Schieberegler bewegt wird (Ereignis: OnChange).
Die Lösung des obigen Gleichungssystems ist gegeben durch
$y = (b - a)/(b*b - a*d)$
$x = (b - d)/(b*b - a*d)$
Stellen Sie mit einer if-Anweisung sicher, dass keine Division durch 0 stattfindet.
In diesem Fall braucht kein neuer Wert für x bzw. y angezeigt werden, und im Feld für die Lösung soll „Division durch 0“ stehen.