Lösungen

1 Das erste Projekt

Fragen

1. Ein Programm in C++Builder hört nicht einfach auf, wenn die letzte Programmzeile ausgeführt wurde. Das Formfenster bleibt weiter bestehen. Erst mit einem Klick auf Schließen oder den Tasten [Alt][F4] ist das Programm abgeschlossen.
2. In C++Builder umfaßt die Erstellung und Bearbeitung eines Programms gleich mehrere Dateien. Deshalb bezeichnet man alles zusammen als Projekt.
3. Im **Formfenster** verteilst und ordnest Du Deine Komponenten aus der Komponentenpalette (bis jetzt hast Du nur die Schaltfläche benutzt). Im **Editorfenster** kümmerst Du Dich um die Methoden, damit das Programm laufen kann. Ohne den Programm- bzw. Quelltext führt nämlich eine Aktion wie z.B. das Klicken auf eine Schaltfläche zu keinem erkennbaren Ergebnis.

Keine Aufgaben

2 Datentypen und Operatoren

Fragen

1. Hauptkomponente ist das Formfenster, kurz `Form` genannt. Daneben sind aus diesem Kapitel erst mal zwei weitere Komponenten bekannt: Die Schalt- und die Anzeigefläche (Button und Label).
2. Ereignisse sind z.B. Mausklicks oder Tastendrücke. Mit Ereignissen lassen sich Methoden verknüpfen, die aktiviert werden, wenn ein Ereignis eintritt: So wird z.B. beim Ereignis `OnButton` die Methode `ButtonClick` aktiviert. Viele Methoden sind mit einem Objekt wie z.B. Formfenstern, Buttons, Labels oder anderen Komponenten verbunden.
3. Der Eigenschaft `Caption` muß ein neuer Wert zugewiesen werden, z.B.

```
Button1->Caption := IntToStr (Random (1000)+1);
```

Aufgaben

1. Einen Versuch findest Du im Projekt mit dem Namen Projct2A.mak.
2. Das Projekt Horoskop.mak bietet Dir ein Gerüstprogramm mit zwölf beschrifteten Buttons, zu denen die `ButtonClick`-Methoden bereits existieren. Nur die drei Fragezeichen mußt Du selbst noch durch passende Vorhersagetexte ersetzen.
3. Dazu schau mal in den Programmtext zum Projekt Mathe3.mak.

3 Bedingungen

Fragen

1. Mit dem Operator "=" (einfaches Gleichheitszeichen) werden Ausdrücke oder Werte zugewiesen, mit dem Operator "==" (doppeltes Gleichheitszeichen) werden Ausdrücke oder Werte verglichen.
2. Der Haken ist, daß auch "zu groß" ausgegeben wird, wenn die Zahl richtig ist: Das Gegenteil (else) von < ist nämlich >=.
3. Für *Zensur* wird ein zufälliger Wert zwischen 1 und 6 erzeugt.

Aufgaben

1. Dazu findest Du einen Vorschlag in Mathe2a.mak.
2. Schau Dir dazu mal das Projekt Zensur2a.mak an.
3. Das *Raten*-Projekt trägt den Namen Zraten3a.mak.

4 Kontrollstrukturen

Fragen

1. Das Programm steckt in einer Endlos-Schleife fest: Weil das Kapital sich nie vermehren kann, wird daraus nie eine Million. Heraus kommt man da nur mit den Tasten [Strg][F2].
2. Über die Eigenschaft `ItemIndex` wird bei einem Listenfeld (ListBox) oder Kombinationsfeld (ComboBox) ermittelt, welcher Eintrag ausgewählt worden ist.

Aufgaben

1. Eine Lösung steht in Drueck1.mak.
2. Schau Dir dazu mal das Projekt HSkop2.mak an. (Allerdings gibt's auch hier nur ein Gerüst.)

5 Komponentensammlung

Fragen

1. Wenn Optionsfelder oder Kontrollfelder in einem Formfenster zu einer Gruppe zusammengefaßt sind, kann man bei Optionsfeldern (RadioGroup/RadioButtons) immer nur eine Komponente einschalten. Bei Kontrollfeldern (ChecKBoxes) dagegen lassen sich beliebig viele Komponenten einer Gruppe aktivieren.
2. Der Standardtyp für Zeichenketten in C++ ist `char*`., ein auch an Windows orientierter Stringtyp. Genau besehen wird damit nur eine Stelle im Arbeitsspeicher Deines PC angezeigt, an der diese Zeichenkette beginnt. Über die Länge wird dabei nichts ausgesagt. Das Ende dieser Art von Strings erkennt man an dem Abschlußzeichen #0. Und für den nötigen Platz im Arbeitsspeicher mußt Du selbst sorgen. Daneben bietet C++Builder aber auch den Typ `String`. Damit wird gleich der benötigte Platz im Arbeitsspeicher Deines PC vereinbart. Auch bei Zuweisungen und Verknüpfungen läßt sich dieser Stringtyp bequemer handhaben.

Aufgaben

1. Dazu findest Du ein Gerüstprogramm im Projekt HSkop3.mak.
2. Die Lösung könnte so wie in Klemp4a.mak aussehen.

6 Aktion Seelenklempner

Fragen

1. Beide Felder dienen der Anzeige von Text. Weil die Anzeigetafel (Panel) dabei von einem sichtbaren Rand umgeben ist, der sich einstellen läßt, kann sie auch ähnlich wie ein Gruppenfeld Komponenten umrahmen und zusammenfassen.
2. Diejenige Komponente, die im Formfenster gerade aktiviert ist, besitzt den Fokus. Beim Programmstart erhält die Komponente den Fokus, die der Eigenschaft `ActivControl` der Klasse `TForm` zugeordnet ist.
3. Eine Textdatei kann z.B. über ein Objekt vom Typ `TStringList` mit `LoadFromFile` geöffnet und mit `SaveToFile` gespeichert werden.
4. Als erstes wird eine Instanz vereinbart, z.B. über `TStringList *EineListe;` Damit das Objekt dann auch funktionstüchtig ist, wird mit `EineListe = new TStringList;` der nötige Platz im Arbeitsspeicher reserviert.

Aufgaben

1. Dazu mußt Du Dir mal die Projektdatei Prozent.mak anschauen.
2. Eine Lösung ist in der Methode `TForm1::Button2Click` des Projekts Klemp7a.mak zu finden.

7 Menüs und Dialoge

Fragen

1. Zuerst legt man ein Symbol für die Komponente `MainMenu` auf dem Formfenster ab. Mit Doppelklick kommt man in den Menüeditor. Dort läßt sich jeder einzelne Eintrag eingeben. Im Objektinspektor lassen sich zusätzlich die Eigenschaften anpassen. Nach Doppelklick auf einen Menüeintrag kann man die damit verknüpfte Methode bearbeiten.
2. Am Rückgabewert der Methode `Execute`: Bei allen drei Objekten ist dieser Wert `true`, wenn der Dialog mit OK beendet wird. Ansonsten gibt `Execute false` zurück.
3. Objekte vom Typ `TStringList` sammeln Zeichenkette zeilenweise, sind also sozusagen Behälter für Strings. Objekte vom Typ `TRichEdit` können einen normalen oder formatierten Text (z.B. mit Schriftart und Absatzformaten) aufnehmen, anzeigen und ausdrucken lassen. (Textformate haben die Kennung TXT und RTF.)

Keine Aufgaben

8 Grafik mit Canvas

Fragen

1. Bei `MoveTo` wird sozusagen eine unsichtbare Linie gezeichnet und bei `LineTo` eine Linie in der aktuellen Zeichenfarbe (`Pen->Color`).
2. Man kann auch `void` als Typ ansehen. Danach ist eine Prozedur eine Funktion vom Typ `void`, die eben »ein Nichts« zurückgibt. In frühen C-Versionen gab es wirklich nur Funktionen, der Typ `void` war dort unbekannt.
3. Probier es aus: Ohne `public` kannst Du auf keine Eigenschaft oder Methode mehr zugreifen, C++Builder meldet »not accessible«.

Aufgaben

1. Die eine Anweisung findest Du im Projekt Grafik6a.mak.
2. Dazu solltest Du mal die Projektdatei Quadrat.mak öffnen.

9 Kapselung und Vererbung

Fragen

1. Daten und Code bzw. alle Eigenschaften und Methoden eines Objekts werden zu einer Einheit zusammengefaßt. Ein Zugriff von außen ist nur über den Namen des Objekts möglich – wenn die Elemente als `public` vereinbart wurden.
2. Wird eine Klasse so vereinbart `class Name : public Vorfahr`, dann verfügt sie automatisch über alle Eigenschaften und Methoden des Vorfahren. Mit einer geeigneten Basisklasse läßt sich so eine komplette Klassenfamilie aufbauen, bei dem die letzte Klasse in der Hierarchie alles erbt, was all ihre Vorfahren an Fähigkeiten zu bieten haben. (Voraussetzung ist immer, daß die Vererbung als `public` vereinbart wurde.)
3. In einer Headerdatei werden die Strukturen von Typen bzw. Klassen vereinbart sowie Methoden und Funktionen deklariert. In der zugehörigen Unitdatei werden globale Variablen vereinbart und die Methoden und Funktionen klar definiert.
4. Über **Datei** und **Neu** wählst Du im Dialogfeld **Neue Einträge** das Symbol für Komponente aus. Dann trägst Du im Dialogfeld **Komponenten-Experte** den Namen der Klasse und des Vorfahren ein. C++Builder erzeugt eine neue Unit mit den nötigen Basisvereinbarungen der neuen Klasse. Nach dem Speichern der Unitdatei steht auch die Headerdatei zur Verfügung. Dort kannst Du weitere Vereinbarungen einfügen. Wird in einer anderen Datei ein Objekt der neuen Klasse vereinbart, muß in dieser Datei eine `#include`-Direktive (mit dem Namen der neuen Headerdatei) einfügt werden.

Aufgabe

1. Eine Möglichkeit findest Du im Projekt Mathe4a.mak.

10 Eigene Komponenten

Fragen

1. Mit der Methode `Hide` kannst Du viele Komponenten vom Formular (scheinbar) verschwinden lassen. Und mit `Show` tauchen sie dann wieder auf. Schau dazu mal in das kleine Projekt ShowHide.mak.
2. Das kommt darauf an: Normalerweise kann die Methode `SetSize` nach einer `private`-Vereinbarung nicht mehr benutzt werden, um von außen (z.B. über ein Formular) die Maße des `Movie`-Objekts einzustellen. Andererseits aber sind die Eigenschaften `Left`, `Top`, `Width` und `Height`, die `TMovie` von `TImage` geerbt hat, öffentlich. Ist `TMovie` als Komponente in der Palette von C++Builder verfügbar, dann kann man die Lage und Maße ja auch direkt einstellen.

Aufgaben

1. Eine Lösungsmöglichkeit findest Du im Projekt MShow.mak. (Funktioniert aber nur, wenn `TMovie` als Komponente in die Palette von C++Builder eingebunden wurde!)
2. Dazu schau mal in die Projektdatei OOP2a.mak.

11 Für alle Fälle MDI?

Frage

1. Weder noch, denn C++Builder arbeitet mit mehreren Fenstern, die zum Teil voneinander unabhängig sind. Man könnte aber sagen: Die Entwicklungsumgebung C++Builder besteht aus einem System von MDI- und SDI-Anwendungen.

Aufgabe

1. Da könnte Dir das Projekt Rtext1.mak weiterhelfen.

12 Virtuelle Methoden

Fragen

1. Die `Click`-Methode wird aufgerufen, wenn die linke Maustaste losgelassen wurde, was nicht in unserem Sinne ist. Denn dann wird der `OButton` zwar eingedrückt, bleibt aber unwiderruflich in diesem Zustand.
2. Bei der frühen Bindung werden alle Aufrufe von Methoden beim Kompilieren eindeutig festgelegt. Bei der späten Bindung bleibt die Aufrufstelle offen. Und erst während der Laufzeit des Programms wird entschieden, welche Methode aufgerufen wird (Polymorphie).
3. Ja, und wie das geht, siehst Du im Projekt FStein2B.mak.

Aufgaben

1. Dazu nimm Dir mal das Projekt FStein3.mak vor.
2. Die `virtual`-Vereinbarungen stehen in der Headerdatei Movie2B.h, das passende Projekt dazu heißt OOP2b.mak.
3. Eine Möglichkeit findest Du im Projekt OOP3a.mak (Obutton1a).
4. Schau mal in die Projektdatei OOP4.mak (Obutton2). Ich habe der neuen Klasse den (sinnigen?) Namen `TOOButton` gegeben – für »OnOffButton«.
5. Ein Versuch ist im Projekt OOP5.mak zu finden. (Funktioniert aber nur, wenn `TOButton` zuvor in der Komponentenpalette von C++Builder installiert wurde!)

13 Polymorphe Klassen

Keine Fragen

Aufgabe

1. Da gibt es einige (ab Kapitel 9). Ein Beispiel ist in der Headerdatei Movie2C.h zu finden, die zugehörige Projektdatei ist OOP2C.mak.

14 Buntes Allerlei

Fragen

1. Bei Eingangsparametern wird deren Wert kopiert und innerhalb der Methode verarbeitet. Außerhalb ändert sich nichts, weil ja mit einer Kopie gearbeitet wurde. Bei Durchgangsparametern wird gar keine Kopie erzeugt, sondern über die Adresse direkt der Wert verarbeitet und damit auch außerhalb der Methode verändert.
2. Angenommen, die Vereinbarung ist `String Name;`. Dann steht in `Name` eine Zeichenkette, und in `&Name` die Adresse im Speicher, an der diese Zeichenkette beginnt. Angenommen, die Vereinbarung ist `String *Name;`. Dann steht in `Name` die Adresse im Speicher, an der eine Zeichenkette beginnt, und in `*Name` die Zeichenkette selbst.

Aufgaben

1. Zugegeben ein bißchen verzwickt, weil man "++" sowohl vor (Präfix) als auch hinter (Suffix) einer Variablen bzw. einem Objekt einsetzen. Um "++" als Suffix-Operator zu kennzeichnen, ist daher ein sogenannter Dummy-Parameter nötig. (Ansonsten gibt C++Builder eine Warnung aus, was aber nicht schlimm sein muß.) Eine Lösung steht im Projekt Fstein3a.mak.
2. Dazu schau Dir mal das Projekt Fstein8a.mak an. (Funktioniert aber nur, wenn `TOButton` als Komponente in die Palette von C++Builder eingebunden wurde!)

3. Die `template`-Funktion und ihren Einsatz findest Du im Projekt Zensur3.mak.