

## Projekt Zahlenraten

Das zu entwickelnde Zahlenrate-Programm soll folgende Eigenschaften besitzen:

- Der Rechner erzeugt selbstständig eine Zufallszahl im Bereich von 1 bis 100.
- Nach jedem Versuch wird die Eingabe vom Programm mit „Die Zahl ist zu groß“, „Die Zahl ist zu klein“ oder „Das war richtig!“ bewertet.
- Nach dem Erraten kann eine neue Zufallszahl erzeugt werden.

### 0. Neues Projekt anlegen:

- Wählen Sie im Menü **Datei / Neue Anwendung**.
- Speichern Sie unter **Datei / Projekt speichern unter**
- Legen Sie in ihrem Delphi-Verzeichnis **H:/delphi/** einen neuen Ordner an: **H:/delphi/zahlenraten**
- Speichern Sie in diesem Ordner
  - die **Unit** unter dem Namen **zahlenratenU.pas** und
  - das **Projekt** unter dem Namen **zahlenraten.dpr**



### 1. Erstellen des Formulars:

- Erstellen Sie eine Form nach nebenstehender Vorlage.
- Wählen Sie die **Bezeichner** der Komponenten (**Eigenschaft Name** im Objektspektor) sinnvoll.
- Verwenden Sie für die Überschrift und das Ergebnisfeld die **Komponente Panel** im **Register Standard**. Diese Komponente erzeugt ein hervorgehobenes Feld, auf dem mehrere weitere Komponenten platziert werden können, welche dann in fester Anordnung zusammen mit dem Panel bewegt werden können. (In diesem Projekt verwenden wir die Komponente ausschließlich als hervorgehobenes Textfeld.)
- Die drei Knöpfe „Neu“, „Raten“ und „Abbruch“ sind **BitButton-Komponenten** (Register Additional) mit den Eigenschaften „**bkRetry**“, „**bkOK**“ und „**bkAbort**“.

## Lokale und globale Variablen

Bisher haben Sie nur **lokale Variablen** verwendet, d.h. die Variablen werden innerhalb einer Prozedur deklariert und sind deshalb **nur innerhalb der jeweiligen Prozedur gültig**. Innerhalb einer anderen Prozedur können diese Variablen nicht verwendet werden. (Wenn man eine Variable mit gleichem Namen in zwei verschiedenen Prozeduren verwendet, so existieren diese beiden Variablen unabhängig voneinander. Änderungen am Wert der Variablen in einer Prozedur wirken sich nicht auf den Wert der anderen Variablen aus.)

In diesem Programm benötigen wir jedoch eine Variable Zufallszahl, die in einer Prozedur (BitBtnRetryClick) erzeugt wird und in einer anderen Prozedur (BitBtnOK) zur Auswertung kommt, d.h. die Variable muß nicht nur innerhalb einer einzelnen Prozedur, sondern **innerhalb des gesamten Programms gültig** sein. Solche Variablen nennt man **globale Variablen**.

#### Globale Variablen

deklariert man direkt am Anfang des Implementation-Teils nach dem Einbinden des Formulars {\$R \*.dfm}:

```
implementation
{$R *.dfm}
var Zufallszahl: Integer;
```

(Bemerkung: Verwendet man innerhalb einer Prozedur eine lokale Variable gleichen Namens, so hat die lokale Variable Vorrang. Die globale Variable ist dann innerhalb dieser Prozedur nicht gültig!!)

## Die Zufallsfunktion Random()

Die Funktion **random(Zahl)**; erzeugt ganzzahlige Zufallszahlen im Bereich 0 bis Zahl – 1. Um Zufallszahlen in einem Bereich zwischen Anfangszahl und Endzahl zu erzeugen, muß man folgenden Quelltext eingeben:

```
Zufallszahl := Random(Ende - (Anfang - 1)) + Anfang;
```

z.B. erzeugt

```
Zufallszahl := Random(6) + 5;
```

Zufallszahlen zwischen  
5 (Anfang) und 10 (Ende)

1. Geben Sie deshalb zur Erzeugung einer **Zufallszahl zwischen 1 und 100** folgenden Quelltext innerhalb der **Prozedur BitBtnRetryClick** ein:

```
procedure TFoZahlenraten.BitBtnRetryClick(Sender: TObject);  
begin  
    Zufallszahl := Random(100)+1;  
    EdEingabe.Text := IntToStr(Zufallszahl);  
end;
```

Durch die Ausgabe der Zufallszahl im späteren Eingabefeld **EdEingabe** können Sie die Generierung der Zufallszahlen überprüfen. Wenn Sie das Programm nun mehrmals starten und sich jeweils die ersten fünf Zufallszahlen merken, werden Sie bemerken, daß nach jedem Programmstart die gleiche Folge von Zufallszahlen erzeugt werden. Damit stellt Delphi dem Nutzer eine reproduzierbare (!!) Anzahl von gut verteilten Zahlen zu statistischen oder zu Testzwecken zur Verfügung. Nachdem Sie sich von dieser Funktion überzeugt haben, können Sie den Quelltext für die Ausgabe der Zufallszahl wieder löschen. (Bemerkung: Ruft man die Zufallsfunktion Random ohne Parameter auf, so werden gebrochene Zufallszahlen im Bereich 0 bis 1 erzeugt.)

2. Um zu garantieren, daß bei jedem Programmstart nicht die selbe Folge von Zufallszahlen erscheint, wird bei Erzeugen der Form (Doppelklick auf Ereignis **OnCreate** der Komponente **FoZahlenraten** im Objektinspektor) die **Prozedur Randomize** aufgerufen und die erste Zufallszahl erzeugt:

```
procedure TFoZahlenraten.FormCreate(Sender: TObject);  
begin  
    Randomize;  
    Zufallszahl := Random(100)+1;  
end;
```

## Überprüfung der Zufallszahl

3. Implementieren Sie die Prozedur **TFoZahlenraten.BitOKClick**, welche die globale Variable **Zufallszahl** mit der momentanen Eingabe **EdEingabe.Text** vergleicht, und je nach Ergebnis dieses Vergleichs eine entsprechende Meldung im Ergebnispanel ausgibt (siehe Aufgabenstellung). Die Ausgabe einer String-Konstante erfolgt durch hochgestellte Kommata. In unserem Beispiel

```
PaErgebnis.Caption := ' Die Zahl ist zu klein! '
```

Als Hilfestellung noch einmal die Syntax für die benötigten **if...then...else...** Entscheidungen:

### Verzweigte Entscheidungen:

```
if {Bedingung_1}  
then {Anweisung_1}  
else  
    if {Bedingung_2}  
    then {Anweisung_2a}  
    else {Anweisung_2b};
```

### Verbundanweisungen

```
if {Bedingung}  
then  
    begin  
        {Anweisung_1a};  
        {Anweisung_1b};  
    end  
else {Anweisung_2};
```

## Zusatzaufgaben:

- Die Überprüfung soll nicht nur durch Mausklick auf die „Rate“-Taste, sondern auch durch **Druck auf die „Enter“-Taste** erfolgen. Weisen Sie dazu der **Eigenschaft Default** des „Rate“-Buttons den Wert **true** zu.
- Nach Eingabe einer Zahl soll das Eingabe-Editierfeld wieder automatisch den **Eingabefokus** erhalten, ohne daß man das Feld immer erst mit der Maus anklicken muß. Fügen Sie dazu am Ende der Prozedur **TFoZahlenraten.BitOKClick** folgende Zeile ein:

**EdEingabe.SetFocus;**

Alle Komponenten mit Eingabefokus (z.B. Button, BitBtn, Edit, SpinEdit, etc.) besitzen diese Methode! Markieren Sie das Wort **SetFocus** und drücken Sie die Funktionstaste **F1**, um sich weitergehende Informationen über diese Methode anzeigen zu lassen. Verwenden Sie diese Hilfestellung so oft wie möglich!

- Falsch geratene Zahlen sollen eine **blaue Aufschrift** auf dem Ergebnispanel erzeugen, richtig geratene Zahlen eine **rote Aufschrift**. Da nun mehrere Anweisungen (Ausgabetext und Textfarbe) auf eine Entscheidung folgen, müssen diese als **Verbundanweisung** (siehe oben) implementiert werden. Die Anweisung zur Änderung der Farbe lautet:

**PaErgebnis.Font.Color := clRed;**

Die Namen der Farbkonstanten können Sie dem Objektinspektor (**Eigenschaft Font.Color**) entnehmen. Informieren Sie sich auch hier durch Markierung der Konstante **clRed** und **F1** über die möglichen Werte.

- Strukturieren Sie Ihren Quelltext, falls noch nicht geschehen, indem Sie zusammengehörende bzw. gleichwertige Teile gleich weit einrücken. Meist erfolgt die Einrückung um **zwei Leerzeichen**. Durch die Tastenkombination **Strg+Shift+I** (ein Zeichen nach rechts) bzw. **Strg+Shift+U** (nach links) können Sie mehrere Zeilen gleichzeitig ein- bzw. ausrücken. Folgende Strukturierung hilft bei späterer Fehlersuche:
  - Zusammengehörende Schlüsselwörter **begin** und **end** werden immer gleich weit eingerückt. Damit lassen sich fehlende oder überzählige Schlüsselwörter leichter erkennen.
  - Der Quelltext zwischen **begin** und **end** wird um zwei Zeichen eingerückt.
  - Bei einer **if...then...else..** Anweisung werden die Schlüsselwörter **then** und **else** gleich weit, aber um zwei Zeichen relativ zur if-Bedingung eingerückt.
  - Ein durch **begin** und **end** geklammerte Verbundanweisung wird ebenfalls eingerückt.
- Erweitern Sie das Programm um ein weiteres Anzeigeelement, auf dem die jeweilige Anzahl der bisher benötigten Versuche angezeigt wird:
  - Wie kann man mit möglichst wenigen Versuchen die Zufallszahl erraten ?
  - Wie viele Versuche sind bei 100, 1000, 10000 bzw. n Zahlen höchstens notwendig ?

## Weitere Aufgaben mit Entscheidungen:

- Lösung lineare und quadratischer Gleichungen:**  
Implementieren Sie die Aufgaben der letzten Woche zur Lösung linearer und quadratischer Gleichungen.
- Schaltjahr:**  
Nach Eingabe einer Jahreszahl soll bestimmt werden, ob es sich um ein Schaltjahr handelt oder nicht. (Tipp: Informieren Sie sich über die arithmetischen Operatoren **div** und **mod** in der Delphi-Hilfe !)
- Body-Mass-Index (BMI):**  
Die moderne Ernährungswissenschaft rechnet mit dem Body-Mass-Index und

**„dividiert die Masse in kg durch das Quadrat der Körpergröße in m.“**

Ihr Programm soll den BMI nach Eingabe von Geschlecht, Körpergröße und Masse berechnen und die entsprechende Gewichtsklasse ausgeben:

| BMI      | männlich | weiblich | Gewichtsklasse         |
|----------|----------|----------|------------------------|
| unter 20 | 20       | 18       | Untergewicht           |
| unter 25 | 25       | 24       | Normalgewicht          |
| unter 30 | 30       | 30       | Übergewicht            |
| über 30  | 30       | 30       | medizinisch bedenklich |