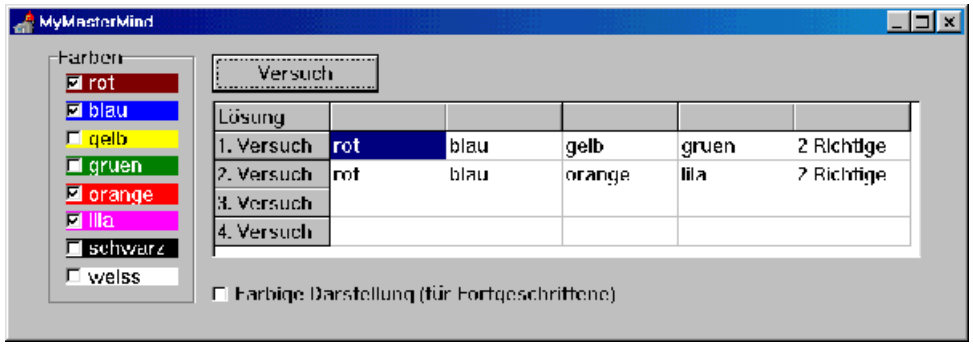
Programmiersprachen 2 — Übung 4: Bonusaufgabe — WS2000

(Mengen, Aufzählungstypen, Checkboxen)

Aufgabenstellung:

Das Spiel MASTERMIND soll in abgewandelter Form programmiert werden.



Regeln:

Der Computer definiert eine Menge mit vier aus acht Farben.
Der Spieler muß versuchen, diese vier Farben in 4 Versuchen zu erraten.
Die doppelte Eingabe von Farben ist nicht zulässig.

Komponenten:

Wählt zur Auswahl der Farben eine *GroupBox*, in der Ihr 8 *Checkboxen* unterbringt. Während die *GroupBox* nur zum optischen Zusammenhalt beiträgt, könnt Ihr die *CheckBoxen* über ihre Eigenschaft *Checked* abfragen/setzen.

Vorgaben:

- Da Mengen Thema der Übung sind, soll natürlich auch mit Ihnen gearbeitet werden. Schreibt deshalb folgende Prozeduren/Funktionen:

```
function AnzahlInMenge (Menge: TMenge): byte;
    {gibt zurück, wieviele Elemente in der Menge sind}
function CheckBox2Menge (VAR Menge: TMenge): boolean;
    {wandelt die CheckBoxen in eine Menge; gibt TRUE, wenn
    Anzahl der angeklickten Boxen richtig war }
procedure Menge2Grid (Menge: TMenge; Zeile: byte);
    {macht entsprechend der Menge eine Ausgabe in die
    übergebene Zeile des Grids}
```

Ablauf:

```
CheckBox in Menge wandeln
IF genau 4 angeklickt THEN
   Menge in Grid übertragen
   IF Richtig Geraten
        Meldung; NeuInitialisieren
   ELSE IF Versuche überschritten THEN
            Lösung ausgeben
            Meldung;
            Initialisieren
         ELSE Versuch:= Versuch+1;
ELSE Fehlermeldung (Falsche Eingabe)
```

Hinweise zu Mengen:

- Mengen können im Deklarationsteil folgendermaßen deklariert werden:

```
TYPE FarbenTyp = (rot, gelb, gruen);
     MengenTyp = SET OF FarbenTyp;
CONST VolleMenge = [rot..gruen];
```

- [] stellt die leere Menge dar
- Ist die Variable *c* vom Typ *char*, so entspricht der Ausdruck *[c]* einer Menge, die nur das in *c* gespeichert Zeichen enthält.
- Es existieren folgende Mengenoperationen:
  - \+ Vereinigung
  - \- Differenz
  - * Schnitt

Komponenten mit Variablennamen ansprechen:

Man kann alle Checkboxen in einer Schleife über ihren Namen ansprechen:

```
CONST FarbStr: array[rot..gruen] of TFarbe = (´rot´,..,´gruen´);
...
FOR Farbe:= rot TO gruen DO      {Löschen aller Kreuze}
   WITH Form1.FindComponent(´CheckBox' + FarbStr[Farbe]) as TCheckBox do
      Checked:= false;
   {setzt CheckBoxRot.checked, CheckboxGelb.checked,... auf false}
```

Die farbige Darstellung braucht nicht implementiert werden!

Farbige Darstellung (muß nicht implementiert werden!):

```
procedure TForm1.RatefeldDrawCell(Sender: TObject; Col, Row: Longint;
  Rect: TRect; State: TGridDrawState);
  {wird etwas in eine Zelle geschrieben, tritt dieses Ereignis auf}
  {die Zellen werden entsprechend ihrer Inschrift eingefärbt      }

var f: FarbenTyp;
    Farbe: TColor;
begin
  if (cBFarbig.checked) and                   {wenn Farbe gewünscht und }
     (Col > 0) and                             { nicht linkeste Spalte und}
     (Col < RateFeld.ColCount-1) then begin   {nicht rechteste Spalte }

        for f:= rot to undef do             {Farbe zu der Inschrift suchen}
            if RateFeld.Cells[Col,Row]= FarbTextArray[f]
            then Farbe:= ColorTArray[f];

            {jede Komponente hat ein Zeichenfeld CANVAS, welches z.B.
             mit Brush (Hintergrund), Pen (Zeichenfarbe) bearbeitet werden kann}

        with (Sender as TStringGrid).Canvas do begin  {Zeichenfeld der Zelle}
           Brush.Color:= Farbe;          {Hintergrundfarbe festlegen}
           FillRect(Rect)                {Zelle ausmalen}
        end {with};

   (*     RateFeld.Cells[Col,Row]:= Text    {Text wieder drübersetzen geht nicht,
                                        da sonst wieder OnPaint aufgerufen würde}
    *)
  end {if Einfärben gewünscht}
end;
```